VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

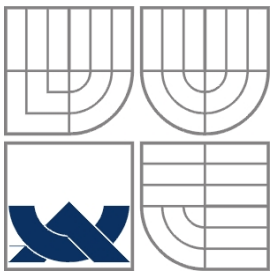
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A SLEDOVÁNÍ MALÝCH POHYBUJÍCÍCH SE OBJEKTŮ

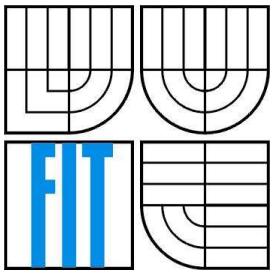
DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE BC. JAN FILIP
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A SLEDOVÁNÍ MALÝCH POHYBUJÍCÍCH SE OBJEKTŮ

DETECTION AND TRACKING OF SMALL MOVING OBJECTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. JAN FILIP

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL HRADIŠ

BRNO 2009

Zadání diplomové práce/6493/2008/xfilip20

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2008/2009

Zadání diplomové práce

Řešitel: **Filip Jan, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Detekce a sledování malých pohybujících se objektů**
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy zpracování obrazu a detekce pohybu.
2. Vytvořte si přehled o současných metodách detekce a sledování pohybujících se objektů ve videu.
3. Navrhněte systém schopný detekovat a sledovat malé pohybující se předměty ve videu ze statické kamery.
4. Vytvořte sadu videí pro experimenty
5. Implementujte navržený systém a proveďte experimenty nad sadou videí.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručný plakát prezentující vaši práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1-3.

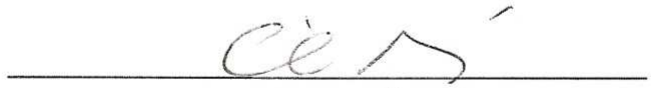
Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hradiš Michal, Ing.**, UPGM FIT VUT
Datum zadání: 22. září 2008
Datum odevzdání: 26. května 2009

L.S.



doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Diplomová práce se zabývá detekcí a sledováním malých pohybujících se objektů ze statického obrazu. Je zde uvedený obecný přehled metod a přístupů řešení detekce a sledování objektů. Dále jsou zde popsány i některé jiné celé přístupy řešení. Jsou zde obsaženy základní definice, jako je šum, konvoluce a matematická morfologie. V práci jsou popsány Bayesovská filtrace a Kalmanův Filtr. Je zde popsána teorie Vlnek, vlnkových filtrů a transformací. Práce se zabývá různými metodami detekce blobů. Je zde uveden návrh a implementace aplikace, který je založen na Vlnkových filtrech a Kalmanově filtru. Implementováno je několik metod odečítání pozadí, které se porovnávají při testování. Testování a aplikace je navržena na detekci vozidel jedoucích v dáli (alespoň 200m daleko).

Abstract

Thesis deals with the detection and tracking of small moving objects from static images. This work shows a general overview of methods and approaches to detection and tracking of objects. There are also described some other approaches to the whole solution. It also included basic definitions, such a noise, convolution and mathematical morphology. The work described Bayesian filtering and Kalman filter. It described the theory of Wavelets, wavelets filters and transformations. The work deals with different ways of the blob`s detection. It is here the design and implementation of applications, which is based on the wavelets filters and Kalman filter. It`s implemented several methods of background subtraction, which are compared by testing. Testing and application are designed to detect vehicles, which are moving faraway (at least 200 m away).

Klíčová slova

Odečítání pozadí, Bayesovský filtr, Kalmanův filtr, Sledování objektů, Šum v obraze, Vlnky, Haarovy vlnky, Daubechiesové vlnky, Matematická morfologie, Konvoluce.

Keywords

Background Subtraction, Bayesian Filter, Kalman Filter, Tracking of Objects, Noise in Image, Wavelets, Haar Wavelets, Daubechies Wavelets, Mathematical Morphology, Convolution.

Citace

Filip Jan: Detekce a sledování malých pohybujících se objektů, diplomová práce, Brno, FIT VUT v Brně, 2009

Detekce a sledování malých pohybujících se objektů

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením Ing. Michala Hradiše. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jan Filip
6.8.2009

Poděkování

Rád bych poděkoval Ing. Michalu Hradišovi za jeho pomoc, ochotu, trpělivost, odborné vedení a připomínky týkající se této práce. Dále pak chci poděkovat Anastasii Pampouchidou z TEI of Crete za její pomoc při shánění kamery. A samozřejmě děkují i mé rodině, která mě podporovala a měla se mnou trpělivost, především mému bratrovi Josefovi za opravy a korektury textu.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Tato diplomová práce se zabývá detekcí a sledováním malých pohyblivých objektů ze statické kamery v reálném čase. Jedná se tedy o zpracování vstupního videa, kde se vyhledává vše, co se pohybuje, a následně je snaha tyto pohyblivé objekty sledovat. To znamená, že v případě sledování více objektů v jeden okamžik, se nemají zaměnit sledované objekty mezi sebou. Z detekcí a sledováním pohybů v obraze pak úzce souvisí i další metody jako jsou odečet pozadí, Bayesovská filtrace a z ní vycházející Kalmanův filtr, které zde budou rovněž popsány.

Využití detekce a sledování pohyblivých objektů v obraze

Nejprve však kde se taková detekce a sledování pohyblivých objektů může uplatnit. Dnes se z detekcí a sledování rozličných objektů setkáme téměř na každém kroku. Od sportu, přes nejrůznější bezpečnostní monitorování až po armádu.

Ve sportu jde většinou o sledování nejrůznějších míčů. Velice populární je například tzv. Jestřábí oko, které se používá v tenise, a které rozhoduje o tom, kam dopadl sporný míček. Jistý typ detekce najdeme třeba i ve fotbale, kde se využívá systém několika kamer, který detekuje a sleduje jednotlivé hráče na hřišti a měří jim uběhnutou vzdálenost za zápas nebo jejich maximální rychlost.

Pak tu máme širokou škálu nejrůznějších bezpečnostních systémů, jejichž cílem bývá hlavně detekovat nějaký přestupek a u těch systému, u kterých to je potřeba i objekt, který přestupek spáchal dále sledovat. Nebo můžeme daný systém použít pro analýzu dopravní situace na daném místě a na základě toho upravit plynulost dopravy, viz [1].

Další oblastí uplatnění se týkají v podstatě zjednodušení práce s počítačem. Na jedné straně se dá využít sledování různých objektů u indexací videí. To znamená automatickou anotaci a vyhledávání videa v multimediální databázi. A na druhé straně se je možné využít detekce sledování pohybu k tomu, aby si člověk otevřel další komunikační kanál pro interakci s počítačem. Jako je rozpoznávání gest, sledování kam se dívají oči, atd.

A pak tu máme široké spektrum využití různorodými armádními systémy. Zde většinou potřebujeme objekty detekovat, sledovat a zneškodnit. Samozřejmě se běžně používají radary, ale ani ty neodhalí vše a pak může přijít na řadu i detekce na základě vizuálního vstupu, užitím kamery.

V mém případě jsem se v rámci semestrálního projektu nejprve zaměřil na detekci malých rychle se pohybujících míčku z jedné statické kamery, jako je míček na stolní tenis nebo větší tenisový míč. Cílem diplomové práce má být detekce objektů o velikosti pouhých pár pixelů. Například detekovat a sledovat v dáli na silnici auta a určit tak například počet projetých automobilů za daný časový úsek.

Přehled práce

Tato práce popisuje nezbytnou teorii, kterou je třeba znát. Dále je zde přehled teoretický i praktický metod, které by měly vést k požadovanému výsledku.

Hned v následující kapitole jsou popsány některé důležité přístupy zpracování obrazu. Začíná se od začátku obrazovou funkcí. Dále přes filtraci, konvoluci a matematickou morfologii. Je zde základní rozdělení metod detekce a sledování. Dále jsou zde popsány některé problémy, se kterými se často musí systémy založené na vizuálním vstupu potýkat, jako je například šum v obraze. Pak je zde popsána problematika Bayesovské filtrace, ze které vychází dále používaný Kalmanův filtr. Na závěr kapitoly je uvedena teorie Vlnek. Společně s ní nejznámější Haarovy vlnky a další typy vlnek, které se dále v práci využívají při návrhu a implementaci řešení.

Ve třetí kapitole se pak zabývám konkrétními metodami a jejich variantami, které jsou základními stavebními prvky celého systému detekce a sledování. Jsou zde popsány metody odečtu pozadí, které detekují pohyblivé části scény od těch statických. Dále je zde popsán Kalmanův filtr a jeho využití v procesu sledování objektu. A na závěr je zde popsána problematika detekce blobů.

Čtvrtá kapitola je přehledem několika vybraných úplných řešení, která se u aplikací detekce a sledování obrazu ze statického obrazu využívají. Já zde nenavrhuji nic výrazně nového.

Pátá kapitola popisuje návrh celého systému, který by měl být dostatečně robustní pro řešenou třídu scén. Je zde vymezená řešená třída scén. Dále tu je celkový návrh struktury aplikace a jednotlivé části jsou dále podrobněji rozvrženy.

V šesté kapitole je poměrně podrobně popsána implementace navrženého řešení. Soustředím se zde na to, jak funguje hlavní řídící funkce a inicializace sledování pohybujícího se objektů. Pak je tu popsána rovnice vlnkové transformace, která má nestarost odečet pozadí.

Návrh experimentů, podpůrných anotačních programů a výsledků je popsán v sedmé kapitole. Jsou specifikovány jednotlivé testy a třídy videí, se kterými se experimentuje. Kapitola obsahuje i grafické ukázky detekce.

V závěrečné kapitole se bilancuje nad dosaženými výsledky a dalšími možnostmi vývoje systému. V příloze A je pak manuál ovládání aplikace a anotačních programů.

2 Základy zpracování obrazu a detekce pohybu

Zde popisuji teoretické základy, které se týkají problematiky této práce. Nejprve obecněji o základech zpracování obrazu. Dále popisuji existující přístupy vedoucí k detekci pohybu a uvádím jejich základní kategorizaci a srovnání. Následně pak popisuji problémy, s kterými se často musíme u obrazu často vypořádat. Hlavně tedy šum a jeho různé varianty. Též je zde popsána problematika Bayesovské filtrace, ze které vychází v problematice detekce a sledování objektů často používaný Kalmanův filtr. Na závěr se pak zabývám teorií vlnek, kterých ve svém přístupu detekce pohybu využívám.

2.1 Obecně o zpracování obrazu

Předmětem zpracování a případného rozpoznávání pohybu v obrazu je obrazová informace o reálném světě, která do počítače vstupuje nejčastěji televizní či jinou kamerou. Cílem je porozumění obsahu obrazu. Postup zpracování a rozpoznávání pohybu v obrazu reálného světa lze podle [33] obvykle rozložit do několika základních kroků. *Snímání, digitalizace a uložení obrazu v počítači*. *Předzpracování obrazu*. *Segmentace a popis pohybujících se objektů*. *Porozumění obsahu obrazu, případně jen klasifikace objektů*.

Prvním krokem zpracování obrazu je *snímání, digitalizace a uložení obrazu* v číselné formě do počítače. Při snímání se převádějí vstupní optické proměnné na spojitý elektrický signál. Vstupní informací může být jas, nebo několik spektrálních složek (*RGB model – Red Green Blue colors*) v případě barevného snímání.

Digitalizací se převádí vstupní spojitý signál do diskrétního tvaru. Vstupní analogový signál je popsán funkcí *f(x,y)* dvou proměnných představujících souřadnice v daném obraze. Funkční hodnota může odpovídat třeba jasu. Vstupní signál je dále vzorkován a kvantován. Výsledně tedy popisuje obraz matice čísel, kdy se jednomu prvku matice říká obrazový element neboli *pixel*.

Druhým krokem je *předzpracování obrazu*. Zde je cílem potlačit šum a zkreslení, které vzniká při digitalizaci a přenosu obrazu. Předzpracováním se také zvýrazňují určité rysy obrazu podstatné pro následné zpracování. Příkladem může být zostření obrazu či jeho vyhlazení.

Třetím a asi nejtěžším krokem postupu zpracování je *segmentace obrazu*, která dovolí v obraze najít pohybující se objekty. Za objekty lze považovat ty části obrazu, které nás z hlediska dalšího zpracování zajímají. Například objekty s periodickým pohybem, jako třeba vodní hladina, nás většinou nezajímají. Popis nalezených objektů v obraze je pak ovlivněn tím, na co se bude užívat. Ten lze popsat dvěma způsoby. Kvantitativně pomocí souboru číselných charakteristik či kvalitativně

pomocí relací mezi objekty. Za velmi jednoduchý popis lze považovat i velikost plochy objektu, i ten je však často dostačující.

Velkou komplikací při zpracování obrazu je vztah mezi jasem, který měří kamera, a tvarem povrchu 3D objektů v obraze. Jas bodu je totiž závislý na velkém množství vlivů - odrazivost povrchu, poloha a vlastnosti zdrojů světla.

2.2 Pojmy základního zpracování obrazu

Problematika zpracování obrazu se opírá o několik základních pojmů, které je dobré znát. Prvním z pojmů je *obraz*. Kdy se obrazem rozumí optický obraz ve standardním slova smyslu. Například může jít o obraz snímaný webovou kamerou.

Obrazová funkce

Formální definice obrazu je popsána matematickým modelem. Jde o spojitou funkci dvou proměnných – *obrazová funkce f(x, y). Jas* bývá nejčastěji hodnotou obrazové funkce, přičemž jas je obrazová veličina, která vyjadřuje vlastnosti signálu obrazu tak, jak obraz vnímá člověk.

Obrazová funkce je *spojitá* nebo *diskrétní*. Spojitá funkce má obor hodnot a definiční obor spojitý. Naproti tomu diskrétní funkce má obor hodnot tvořen množinou diskrétních bodů a jejím oborem hodnot je diskrétní množina.

Samotný obraz je *monochromatický* nebo *multispektrální*. V prvním případě jde o obraz, který je reprezentován jedinou obrazovou funkcí. Příkladem monochromatického obrazu může být *černobílý* obraz nebo obraz v *barvě šedi*. Multispektrální, nebo také barevný obraz, se skládá z dvojic souřadnic obrazu, kterým odpovídá vektor hodnot, který obsahuje jasy jednotlivých barevných složek v obraze. Barevná složky pak bývají často reprezentovány trojicí jasů červeného, zeleného a modrého světla *(RGB – red, green, blue)*.

Spojitá obrazová funkce se pro reprezentaci v počítači diskretizuje na digitální obraz. Digitalizací je míněn postup, kdy se *vzorkuje* obraz maticí bodů a *kvantuje* spojitá úroveň jasové funkce. Čím jemnější je potom vzorkování a kvantování, tím přesnější je převod původního spojitého obrazu na digitální.

Konvoluce

Konvoluce je důležitou lineární operací zpracování obrazu. Konvoluce je definována integrálem

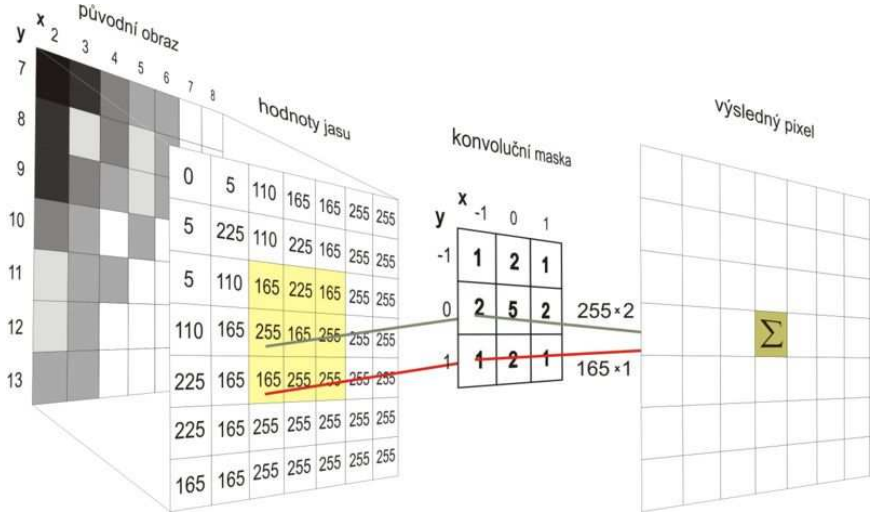
$$f(x,y) * h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-a, y-b)h(a,b)\mathrm{d}a\mathrm{d}b \tag{2.2}$$

nad dvourozměrnými spojitými funkcemi *f* a *h*, kde funkce *h(x, y)* se nazývá *konvolučním jádrem*.

U zpracování digitálního obrazu se používá diskrétní varianta konvoluce, tzv. *diskrétní konvoluce*, která je diskrétní dvojrozměrnou podobou vztahu 2.2

$$I(x,y) * h(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x-j, y-j)h(i,j). \tag{2.3}$$

V tomto případě je *I(x, y)* diskrétní obraz a *h(x, y)* je jádro konvoluce [33]. Konvoluční jádro můžeme chápat jako tabulku (konvoluční maska), kterou položíme na příslušné místo obrazu. Každý pixel překrytý tabulkou vynásobíme koeficientem v příslušné buňce a provedeme součet všech těchto hodnot. Tím získáme nový pixel (viz. Obrázek 2.1)



Obrázek 2.1 Princip výpočtu dvourozměrné diskrétní konvoluce [17].

Filtrace obrazu

Další částí fáze předzpracování obrazu je *filtrace*. Filtrace shrnuje metody, které využívají k výpočtu jasu bodu ve výstupním obraze pouze lokální okolí odpovídajícího bodu ve vstupním obraze. K filtraci se často využívá konvoluční jádro, které většinou definuje použité okolí. Pří výběru okolí se většinou vybírá pravoúhlé okolí, které má nejčastěji velikost rovnu lichému přirozenému číslu. To proto, aby byla zachována symetričnost vůči středovému elementu okolí.

Filtry se používají k vyhlazování, detekci hran či čar, zaostření, atd. Filtr *vyhlazování obrazu* potlačuje vyšší frekvence obrazové funkce. Cílem je potlačit náhodný šum. Přitom ale také dochází k potlačení ostatních náhlých změn jasu, tedy k rozmazání hran. Příkladem jednoduchého vyhlazování může být třeba obyčejné *průměrování*. Zde se každému bodu přiřadí novy jas, který se bere z průměru jasů ve zvoleném okolí. I proto se často zvyšuje váha středového bodu konvolučního jádra či jeho sousedů. Tím se pak dají lépe aproximovat vlastnosti šumu, který má Gaussovské rozložení. Příkladem často používaných průměrovacích filtrů jsou konvoluční jádra

$$h_{16} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}, h_{17} = \frac{1}{17}\begin{bmatrix}1 & 1 & 1\\ 1 & 9 & 1\\ 1 & 1 & 1\end{bmatrix}, h_{9} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}.$$

Příkladem protichůdné filtrace jsou Gradientní operace a s nimi související ostření, které vede ke zdůraznění vyšších frekvencí. V tomto případě jsou zvýrazněny ty části obrazu, kde se jasová funkce náhle mění. Výsledkem tedy je zvýraznění hran v obraze, společně se zvýrazněnými šumovými body. Příkladem takového filtru je maska

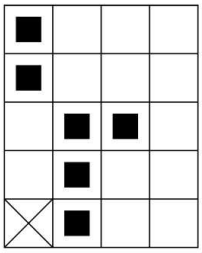
$$h_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

Matematická morfologie – Dilatace a eroze

Dá se říci, že dilatace a eroze jsou také tzv. filtry. Jde však o základní operace *Matematické morfologie*, které mají nejen v předzpracování obrazu široké uplatnění. Využívá se i finálních úprav obrazu a to k detekci hran, segmentaci obrazu, granulometrii a jiné. K této a následující teorii čerpám a cituji z [19] a [20].

Matematická morfologie je teorie a technika pro analýzu geometrických konstrukcí. Je založena na základě terminologie teorie množin. Jedná se o nelineární operaci, která se používá nejčastěji při zpracování obrazu. Používá se i u zpracování grafů a mnohých prostorových struktur. Matematická morfologie byla vyvinuta původně pro binární obraz, později byla rozšířena i na funkce a obrázky ve stupních šedi. Základem matematické morfologie je *bodová množina* a *morfologická transformace*.

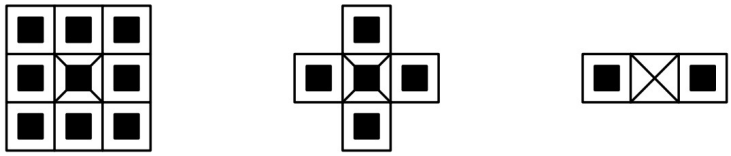
Bodová množina je vlastně digitálním protějškem euklidovského prostoru. Obrázky pak lze modelovat pomocí bodových množin libovolné dimenze. Například 2D euklidovský prostor $\mathbb{E}^2$ a systém jeho podmnožin je přirozeným definičním oborem pro popis rovinných útvarů. Podle množin rozlišujeme binární matematickou morfologii, kdy je množina dána dvojící čísel ($\in \mathbb{Z}^2$, viz. Obrázek 2.2), a šedotónovou matematickou morfologii, ta se skládá z množiny trojic celých čísel ($\in \mathbb{Z}^3$).



$$X = \{(1,0),(1,1),(1,2),(2,2),(0,3),(0,4)\}$$

Obrázek 2.2 Příklad binární bodové množiny [19].

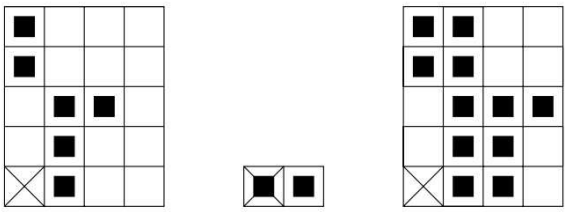
Morfologická transformace je dána relací mezi *obrazem* (obraz označme jako bodovou množinu *X*) a typicky menší bodovou množinou, tzv. *strukturním elementem B* (ukázka na Obrázku 2.3). Strukturní element *B* je vztažen k lokálnímu počátku. Aplikace morfologické transformace na obraz *X* spočívá v systematickém posunu strukturního elementu *B* po obraze. Výsledek transformace v každé poloze odpovídá použité relaci [20].



Obrázek 2.3 Typické strukturní elementy [19].

Jak už bylo popsáno výše *binární matematická morfologie* má definiční obor $\mathbb{Z}^2$, oborem hodnot je $\{1,0\}$ a pracuje se zde s binárním obrazem. Má dvě základní operace *dilatace* a *eroze.* Tyto operace nejsou vzájemně invertovatelné.

$$\begin{aligned} X &= \{(1,0),(1,1),(1,2),(2,2),(0,3),(0,4)\} \\ B &= \{(0,0),(1,0)\} \\ X \oplus B &= \{(1,0),(1,1),(1,2),(2,2),(0,3),(0,4), \\ &\quad (2,0),(2,1),(2,2),(3,2),(1,3),(1,4)\} \end{aligned}$$



Obrázek 2.4 Příklad jak funguje binární dilatace [19].

Binární dilatace, která se značí operátorem $\oplus$, sčítá dvě bodové množiny:

$$X \oplus B = \{p \in \mathbb{E}^2 : p = x + b, x \in X \wedge b \in B\}.$$

To lze vyjádřit i jako sjednocení posunutých obrazů *X*:

$$X \oplus B = \bigcup_{b \in B} X_b$$

Binární dilatace se používá k zaplnění malých děr a úzkých zálivů v objektech(viz. Obrázek 2.5). Dilatace zvětšuje původní velikost objektů. Velikost však lze zachovat tím, že se aplikuje dilatace společně s erozí [20]. O tom více dále.

Obrázek 2.5 Ukázka binární dilatace se strukturním elementem 3x3 (vlevo originál, vpravo po dilataci) [19].

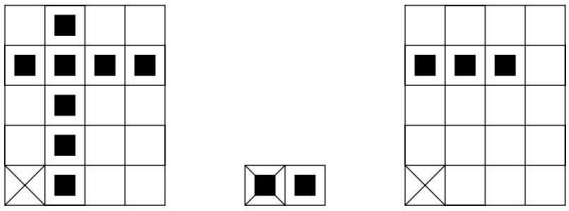
Binární eroze, která se značí operátorem $\ominus$, je duální morfologickou operací vzhledem k dilataci:

$$X \ominus B = \{p \in \mathbb{E}^2 : p + b \in X \text{ pro každé } b \in B\}.$$

Pro každý bod p v obraze se ověřuje, zda pro všechny možné varianty $p + b$ leží výsledek v X. Pokud tomu tak je, pak je výsledek 1, v opačném případě to je 0. Erozi lze ale také vyjádřit jako průnik všech posunů obrazu X o vektory $-b$:

$$X \oplus B = \bigcap_{b \in B} X_{-b}$$

$$\begin{aligned} X &= \{(1,0),(1,1),(1,2),(0,3),(1,3),\\ &\quad (2,3),(3,3),(1,4)\} \\ B &= \{(0,0),(1,0)\} \\ X \ominus B &= \{(0,3),(1,3),(2,3)\} \end{aligned}$$



Obrázek 2.6 Příklad ukazující jak funguje binární eroze [19].

Binární eroze se využívá ke zjednodušení struktury objektu, který je rozložen na jednodušší části. Mimo jiné objekty menší než strukturní element vymizí (viz. Obrázek 2.7). Odečtením výsledků eroze od původního obrazu pak můžeme dostat obrysy daných objektů ve zpracovávaném obraze.

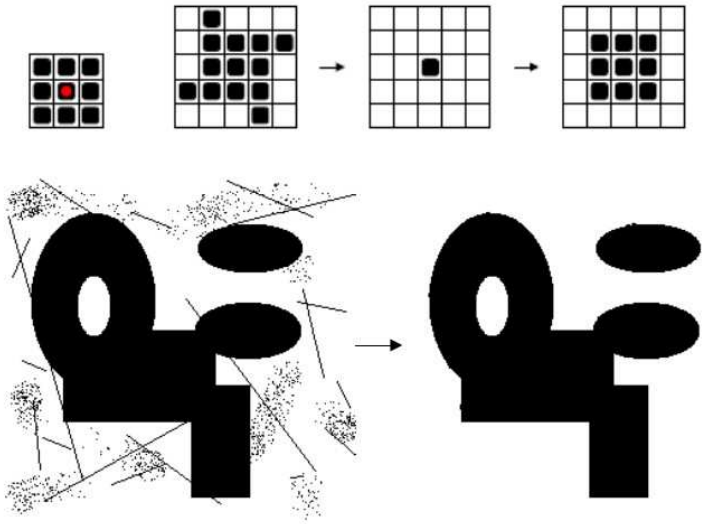
Obrázek 2.7 Ukázka binární eroze se strukturním elementem 3x3 (vlevo originál, vpravo po erozi) [19].

Mezi další operace binární matematické morfologie patří *binární otevření* a *uzavření*. Jde o operace, kde se kombinují eroze a dilatace. Tyto operace se aplikují na obraz pouze jednou, další aplikace těchto operací už další změny nepřináší.

Binární otevření je operací, u které erozi následuje dilatace:

$$\text{Otevření} = (X \ominus B) \oplus B.$$

Pokud se obraz *X* nezmění po otevření strukturním elementem *B*, tak říkáme, že obraz *X* je otevřený vzhledem k *B*. Na Obrázku 2.8 je ukázka, jak operace otevření funguje. Jak je vidět operace otevření odstraňuje šum v obraze.

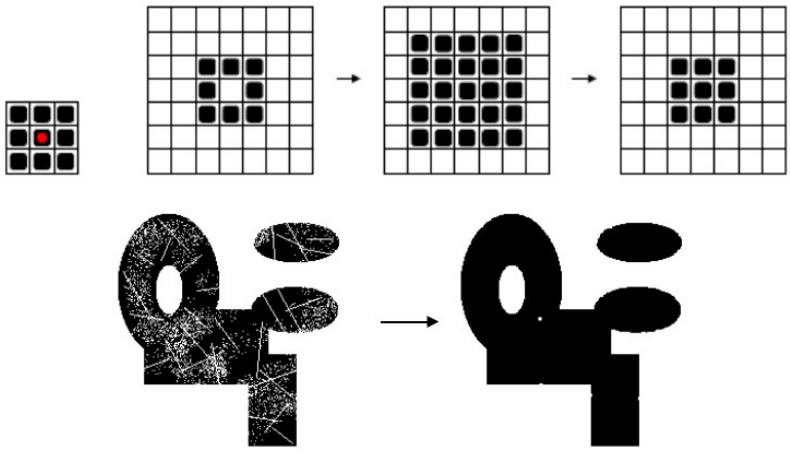


Obrázek 2.8 Příklad binárního otevření [20].

Binární uzavření je operace, kde je dilatace následována erozí:

$$\text{Uzavření} = (X \oplus B) \ominus B.$$

Stejně jako u binárního otevření platí, že pokud se obraz *X* nezmění po uzavření strukturním elementem *B*, tak říkáme, že obraz *X* je uzavřený vzhledem k *B*. Jak je vidět na Obrázku 2.9, tak operace uzavření zaplňuje malé díry a úzké zálivy v objektech [20].



Obrázek 2.9 Příklad operace uzavření [20].

2.3 Přístupy k detekci a sledování objektů v obraze

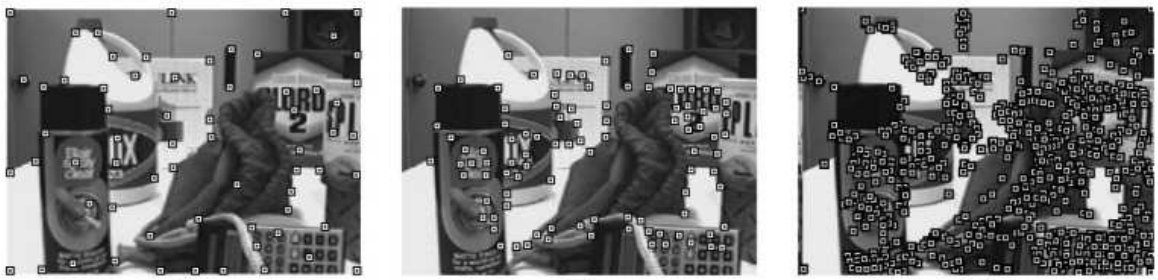
Cílem těchto metod je detekovat a sledovat objekt, který se pohybuje nezávisle vzhledem k pozadí. Přičemž musíme vzít v úvahu dvě situace. Situaci, kdy máme statické pozadí, nebo pohybující se

pozadí (většinou v důsledku pohybu kamery). Tato práce se zabývá sledováním a detekcí objektů ze statického pozadí, takže se budu dále zabývat statickým pozadím. Při kategorizaci metod detekce a sledování pohyblivých objektů je vhodné od sebe oddělit metody používané pro samotnou detekci a naproti tomu pro sledování objektů [16].

Metody detekce objektů

Detekce objektů je základem pro jakékoliv další sledování, protože metody, které se zabývají detekcí objektů, nám podávají základní informaci o tom, že v obraze je něco ke sledování. Tyto metody se provádí nad videem, v jeho každém snímku a nejen na počátku. Metody se obecně liší v tom, jak dlouhou sekvenci snímků k detekci objektů využívají. Běžným postupem je využití sekvence jednoho, dvou snímku. Nicméně pro snížení chybovosti detekce je lepší využívat delších sekvencí. Metody detekce objektů můžeme klasifikovat do následujících 4 kategorií. *Detekce bodů*, *segmentace*, *modelování pozadí* a *supervisor klasifikátory*.

Detektory bodů hledají v obraze důležité body, které nějak reflektují obsah obrazu. Tyto důležité body se pak používají v kontextu řešení pohybových, prostorových nebo sledovacích problémů. K detekci bodů se v literatuře často používá ***Moravcův operátor***. Dále pak ***Harrisův detektor*** a ***KLT***, které jsou podobné. KLT akorát obsahuje další kritérium, které prosazuje předefinované prostorové vzdálenosti mezi zjištěnými důležitými body. A na konec metoda ***SIFT*** (Scale Invariant Feature Transform). SIFT se skládá ze čtyř kroků a generuje podstatně více důležitých bodů ve srovnání s ostatními metodami (viz. Obrázek 2.10). SIFT podává lepší výkony než většina detektorů bodů a je odolný vůči deformacím obrazu.



Obrázek 2.10 Detekce důležitých bodu aplikací a) Harrisova detektoru, b) KLT, c) SIFT [16].

Metody založené na *modelování pohybu* vycházejí z odchylek a změn jednotlivých snímků scény. Pixely, jejichž regiony se mění, se pak dále zpracovávají. Obvykle je zde algoritmus, který spojuje jednotlivé regiony do korespondujících objektů. Tyto metody se pak někdy nazývají jako odečet pozadí.

Mezi tyto metody patří ***Mixture of Gaussians***, který funguje na základě zpracování barevných informací. Dále to je ***Wall flower***, kde se kromě informace barvy využívají prostorové informace, kdy je pixel porovnáván nejen s korespondujícím pixelem v modelu pozadí, ale také se svým okolím. Tato metoda se tak je schopná vyrovnat s třesoucí se kamerou nebo drobnými pohyby v pozadí. Jinak řeší

problematiku metoda ***Eigenbackground***. Ta místo modelování různých individuálních variací pixelů využívá holistického přístupu s využitím *eigenspace* dekompozice. Metody založené na ***Dynamic texture background*** jsou schopné se vyrovnat s jevy, jako mohou být vlny ve vodě, pohybující se mraky a eskalátory. Tyto metody modelují regiony jako ***ARMA*** (*autoregressive moving average*), který poskytuje způsob jak se učit a předpovídat vzory pohybu ve scéně.

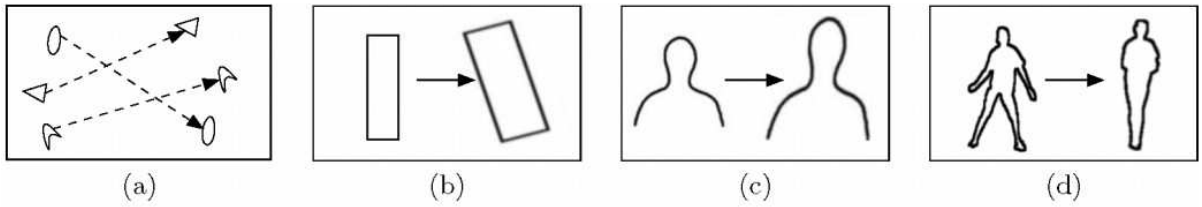
Segmentační algoritmy dělí obraz do percentuálně podobných regionů. Zde bych uvedl pár algoritmů, které jsou relevantní k detekci a sledování objektů. Jsou to ***Mean-Shift Clustering***, který využívá *mean-shift* přístupu k nalezení shluků, které jsou si podobné v prostoru i v barvě. Metoda se využívá v detekci hran a i právě ve sledování objektů. Dále ***Image Segmentation Using Graph-Cuts***, kde se k segmentaci využívá graf (obrázek) a podgrafy (regiony). Metoda má nevýhodu velkých paměťových nároků, ale oproti předchozí metodě (*Mean-Shift*) potřebuje manuálně nastavit jen pár parametrů. Patří sem i ***Aktivní kontury***, kde segmentace objektu je dosaženo umístěním obrysu na hranice regionu, přičemž vývoj kontury se řídí funkční energii.

Supervised učení je metodou, kde se výsledků dosahuje pomocí učících se mechanismů. Tyto metody nejsou limitovány pouze na neuronové sítě. Učící se metody počítají tzv. nadplochu prostoru, která od sebe odděluje jednotlivé třídy objektů ve více dimensionálním prostoru. Mezi nevýhody *supervised* učících se metod patří fakt, že tyto metody často vyžadují velký soubor vzorků od každé třídy objektu. Navíc tento soubor vzorků musí být ručně anotován. I to se však dá řešit tak, že se použijí dva klasifikátory, které používají malé kolekce anotovaných dat. Tyto klasifikátory používají vzájemně nezávislé příznaky. Po jejich natrénování je každý klasifikátor použit k přiřazení neoznačených dat k trénovací sadě druhého klasifikátoru. Tím se dají získat velmi přesná klasifikační pravidla.

Patří sem metoda ***Adaptive Boosting***, která je iterativní metodou s velmi přesným klasifikátorem. Ten je kombinován mnoha jednoduchými klasifikátory (tzv. slabé klasifikátory), které samy o sobě mají průměrnou přesnost. V kontextu detekce objektů pak může jít o sadu prahů, které jsou aplikovány na obraz. Další metodu je ***Support Vector Machines***, kde jsou data tříděna do dvou tříd hledáním maximální okrajové nadroviny, která odděluje jednotlivé třídy od sebe.

Metody sledování

Cílem trackeru objektu je generovat trajektorii daného objektu, pomocí lokalizace pozice v každém snímku videa. Sledování většinou probíhá tak, že na začátku se během několika snímků s pomocí detektoru objektů utvoří korespondence daného objektu mezi jednotlivými snímky. Později se pak už na základě této korespondence a sledovacího algoritmu odhaduje budoucí pozice sledovaného objektu. Na základě toho jestli je objekt popsán jako bod nebo nějaký geometrický tvar pak u objektu můžeme sledovat pohyb nebo deformaci. Metody sledování objektu si znovu rozdělíme do několika kategorií. A to na *sledování bodu*, *sledování jádra* a *sledování siluety* (Obrázek 2.11).



Obrázek 2.11 Různé přístupy sledování, a) sledování bodu, b) sledování jádra, c) d) sledování siluety [16].

U sledování bodu jsou objekty v po sobě jdoucích snímcích zastoupeny body. Asociace bodů je pak založena na předchozím stavu objektu. Tento přístup vyžaduje vnější mechanismus pro detekci objektů v každém snímku. Patřím sem například **Kalmanův filtr**, ***MGE tracker***, ***GOA tracker*** atd.

U sledování jádra se odkazuje na tvar objektu a jeho výskyt. Například jádro může být obdélníková šablona nebo eliptický tvar s přidruženým histogramem. Objekty jsou pak sledovány výpočtem pohybu jádra v po sobě jdoucích snímcích. Pohyb je většinou ve formě parametrické transformace jako translace, rotace nebo affiní transformace. Patří sem metody **KLT**, ***Layering***, ***Eigentracking*** atd.

Metoda sledování siluety využívá informací zakódovaných do oblasti uvnitř objektu. Tyto informace mohou být ve formě hustoty výskytu a tvaru modelů, které obvyklé mají podobu hranové mapy. Modely objektů a siluety jsou sledovány pomocí shody tvaru nebo vývoje obrysu. Patří sem ***State space models***, ***Variational methods***, ***Heuristic methods***, ***Hausdorff***, ***Hough transform*** atd.

2.4 Typické problémy zpracování obrazu

Při zpracování obrazu se často musíme vypořádat s tím, že obraz je nějakým způsobem degradován. Nejčastěji se jedná o různé typy šumu, optická zkreslení nebo o rozmazání obrazu [8] v důsledku pohybu v obraze nebo pohybu kamery, která obraz snímá. Dalšími problémy, které se vyskytují, konkrétněji u samostatné detekce objektů ve statickém obraze, jsou rychlé změny světelných podmínek, nežádoucí pohyby na pozadí způsobené větrem, stíny a fakt, že některé jednodušší algoritmy detekce objektu vyžadují pro inicializaci obraz bez pohyblivých objektů.

Šum

Šum bývá nejčastější problém (vada) obrazu. Šum vzniká při snímání, přenosu a zpracování obrazu. Přičemž zdroje šumu jsou různé. Může jít o defekty optických soustav, nelinearitu elektrických senzorů, granularitu filmového materiálu, ovlivnění tepelným šumem, špatné zaostření nebo jednoduše o zaokrouhlování při zpracování. Šum můžeme dělit podle několika kritérií. A to na Nezávislý a Závislý šum. Nebo na Adiptivní a Multiplikativní šum. Velikost Nezávislého šumu nezávisí na velikosti obrazového signálu a vzniká například při přenosu kanálem. Aditivní šum pak není závislý na obraze.

Obrázek 2.12 Z levé strany: Bílý šum, Gaussův Adiptivní šum, Úzkopásmový šum, "Pepř a sůl" šum [8].

V obraze můžeme najít různé typy šumu (Obrázek 2.12). Při jejich odstranění je pak dobré vědět o nich co nejvíce. Může to být Bílý šum, který je vlastně náhodným nekorelovaným šumem, kdy jeho intenzita se s měnící frekvencí nemění. Dále Růžový šum, jehož intenzita rovnoměrně klesá s rostoucí frekvencí, Gaussův šum, který postihuje všechny pixely obrazu a jeho amplituda má Gaussovo rozdělení pravděpodobnosti. Šum „Pepř a sůl“ často nazývaný impulsním šumem. U tohoto šumu jsou postiženy pouze ojedinělé části obrazu, přičemž jde o extrémní hodnoty. A nakonec Úzkopásmový šum, který postihuje pouze některé frekvence obrazu.

Chceme-li daný šum odstranit, je dobré vždy vědět o daném šumu maximum informací, tedy jaký je jeho typ a další jeho charakteristiky v závislosti na typu. Úzkopásmový šum lze například řešit filtrací ve frekvenční oblasti, kdy ve spektru obrazu šum jednoduše ořízneme. Nevíme-li či neznáme-li typ šumu, tak jsme nuceni použít některé z obecných řešení. To může být průměrování z více snímku (a jeho obměny). Zde je však problém, že právě potřebujeme více snímků tytéž scény. Nebo můžeme užít průměrování z jednoho snímku na základě okolí postiženého pixelu.

Průměrování z jednoho snímku na základě okolí postiženého pixelu se pak může dělit na další podtřídy podle užitých metod. To jsou obyčejné průměrování (vyhlazování, průměrování Gaussovou funkcí), rotační masky (z několika masek se vybere ta s nejmenším jasem rozptylu), narůstání okolí a medián.

Rozmazání obrazu a jeho optická zkreslení

Optické zkreslení vzniká z důvodu vady na optice, či geometrického zkreslení, které může například vzniknout na družicovém snímku vlivem zakřivení země. Dané vady se nejčastěji korigují pomocí korekční optiky či pomocí vhodné dekonvoluce.

K rozmazání obrazu většinou dochází při pohybu objektu vzhledem ke kameře. Známe-li přesně konstantní rychlost, směr pohybu obrazu a dobu otevření závěrky kamery, tak můžeme lehce analyticky poruchu opravit. Neznáme-li však všechny informace o zkreslení obrazu, tak se používá Inverzní filtrace (dekonvoluce). Na základě známých parametrů zkreslení jsem pak schopni daný obraz rekonstruovat.

Další problémy

Sem patří problémy, které se vyskytují ve statickém obrazu při samostatné detekci objektů. Tyto problémy se řeší většinou v rámci samotných algoritmů detekce nebo tím, že se snažíme vyvarovat daným problémům při nasazení algoritmu.

Vezme-li se problém rychlé změny světelných podmínek, tak by mělo být snahou využívat algoritmus pouze, když není zvětšená oblačnost, nebo použít algoritmus, který dostatečně rychle umí reagovat na globální změny, třeba s využitím flexibilního adaptivního prahu.

Další problém nežádoucích pohybů na pozadí způsobené větrem lze řešit tak, že se části obrazu s nežádoucími pohyby ignorují, nebo se využije v algoritmu skutečnosti, že pohyby způsobené větrem, jsou většinou periodické.

Pokud jde o problémy se stíny a různými odrazy od povrchu, tak řešení těchto problému bývá většinou mnohem komplikovanější.

2.5 Bayesovská filtrace

Bayesovská filtrace je pravděpodobnostní technika pro datový tok. Technika přesně kombinuje matematickou formulaci systému se sledováním systému. Pravděpodobnosti se používají k reprezentaci stavu systému, *likelihood* funkce potom reprezentují jejich vztah. V této formě se pak aplikuje Bayesovská hypotéza, podle které se pak odvodí s ní související pravděpodobnost [5].

Samotná Bayesovská filtrace je založena na Pravděpodobnostní teorii, Bayesova teorému a Bayesovské inference. Kalmanův filtr, který je dále popsán, a já jej využívám, pak patří do třídy Bayesovských filtrů.

Bayesův teorém

Bayesův teorém [4] (také známý jako Bayesovo pravidlo) byl objeven Thomasem Bayesem v roce 1763 a souvisí s teorií pravděpodobnosti a vysvětluje podmíněné a okrajové rozdělení pravděpodobnosti u náhodných proměnných. V některých případech je pak schopen Bayesův teorém říci jak aktualizovat či opravit proměnou na základě nových pozorování.

Obecně je teorém platný ve všech běžně užívaných interpretacích pravděpodobnosti. Hraje však zásadní roli pokud jde o rozdíl mezi *frequentist* [2] a *Bayesian* [3] pravděpodobností.

Kde *frequentist* přiřazuje pravděpodobnost náhodným událostem na základě frekvence jejich výskytu. Naproti tomu *Bayesian* je využit k řešení problémů s jistou mírou nejistoty. Důsledkem tedy je, že se Bayesovů teorém je více používán u *Bayesian* pravděpodobnosti.

Formulace Bayesova teorému

Bayesův teorém (rovnice 2.1) se týká předpokladu a mezní pravděpodobnosti událostí A a B, kde B má nenulovou pravděpodobnost.

$$\boldsymbol{P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}}, \tag{2.1}$$

Kde každý člen rovnice má své jméno a význam. $P(A)$ je „prior“ nebo mezní pravděpodobnost události A, „prior“ znamená v tom smyslu, že nebere v potaz žádnou informaci z události B. $P(A|B)$ je podmíněná pravděpodobnost události A na základě události B. $P(B|A)$ je podmíněná pravděpodobnost události A na základě události B. $P(B)$ je „prior“ nebo mezní pravděpodobnost události B a chová se jako normalizovaná konstanta.

Bayesovská inference

Bayesovská inference je statická inference, ve které důkaz (rozuměj měření) nebo pozorování jsou užity k obnovení nebo vytvoření nové pravděpodobnosti, dle které může být hypotéza pravdivá.

Bayesovská inference využívá aspektů vědeckých metod jako je zahrnutí sběru důkazů, což znamená, že se zjistí, zda je daná hypotéza konzistentní nebo není. Podle toho jak se pak akumulují důkazy (měření), tak se mění stupeň důvěryhodnosti hypotézy. S dostatkem důkazů by měla být důvěryhodnost, nebo chceme-li pravděpodobnost, buď velmi vysoká, nebo velmi nízká. Bayesovská inference tedy může být použita k rozhodování mezi konfliktními hypotézami. Jedné, která má vysokou pravděpodobnost, by měla být přijata jako pravdivá, a obráceně. Avšak faktem zůstává, že tato metoda může být zkreslená v důsledku inicializačních důvěryhodností (předpokladů), které jsou určeny předtím, než jsou získány a sebrány jakékoliv důkazy.

Metoda využívá numerických odhadů stupně důvěryhodnosti hypotézy předtím, než mohla být provedena měření a počítá stupeň důvěryhodnosti po jejich změření. Tento proces je každým novým měřením opakován. Bayesovká inference většinou spoléhá na stupeň důvěryhodnosti nebo subjektivní pravděpodobnosti na začátku procesu a proto nezbytně nemusí zajistit objektivní metodu výpočtu [6].

2.6 Teorie vlnek

Vlnky jsou matematickým nástrojem určené k hierarchické dekompozici funkcí. Vlnky dovolují funkci hrubě popsat její celkový tvar a její detaily, které mohou být obsáhle nebo taky téměř žádné. Nehledě na to zda vlnky reprezentuji obrázek, křivku nebo povrch, tak dnes představují elegantní techniku pro reprezentaci *LOD (Level of detail)*. Vlnky mají sice své kořeny v aproximační teorii a ve zpracování signálu. Poslední dobou se však hojně využívají v počítačové grafice. Jejich využití je ve *Zpracování a kompresi obrazu, Globálním osvětlení, Hierarchickém modelování, Animacích*, atd.

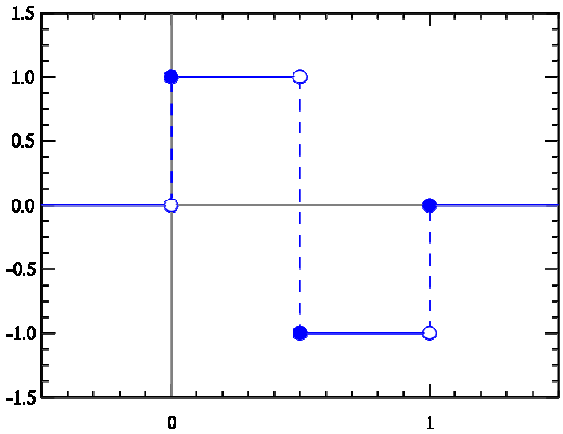
Vývoj vlnek byl motivován hlavně kvůli nutnosti mít rychlé algoritmy, které by pracovaly s kompaktními reprezentacemi funkcí a sadami dat. Vlnkové transformace jsou pak reprezentovány pomocí vlnek. Vlnková transformace má oproti tradičním Fourierovým transformacím řadu výhod u funkcí, které obsahují ve svém průběhu prohlubně a ostré vrcholy, a které jsou reprezentovány konečnou dekonstrukcí a rekonstrukcí a mají neperiodický a nestacionární průběh signálu [14]. Vlnková transformace totiž umí na rozdíl od Fourierovy transformace nejen rozpoznat, jaké frekvence se ve zkoumaném signálu nacházejí, ale také jejich umístění v čase. K popisu jak vlnky fungují, je pak nejvhodnější použít jejich nejjednodušší a nejznámější reprezentaci, Haarovy jedno-dimenzionální vlnky.

Haarovy vlnky

Haarova vlnka je posloupnost funkcí (bude uvedeno dále) a první známou *vlnkou*. Tato posloupnost funkcí byla navržena v roce 1909 Alfrédem Haarem. Haarova vlnka je také nejjednodušší možnou vlnkou (průběh vlnky, viz. Obrázek 2.13). Její nevýhodou však je, že není spojitá a proto není diferencovatelná [22]. Haarova vlnka se skládá stejně jako všechny ostatní vlnky z tzv. *mateřské vlnkové funkce* (*mother wavelet fuction*) a *rozsahové funkce* (*scaling function*). Tyto funkce nabývají následujících hodnot:

, kde , kde

kde *j* je měřítko funkce a *i* posun.



Obrázek 2.13 Haarova vlnka [22].

Haarova báze má důležitou vlastnost zvanou ortogonalita, tuto vlastnost nemají všechny ostatní vlnkové báze. Haarova báze je také nelineární. Další vlastností Haarovy báze je, že je jí možné normalizovat. Normalizace Haarovy báze je možná po úpravě její definice na

,

$$\psi_i^j = 2^{j/2}\psi(2^j x - i),$$

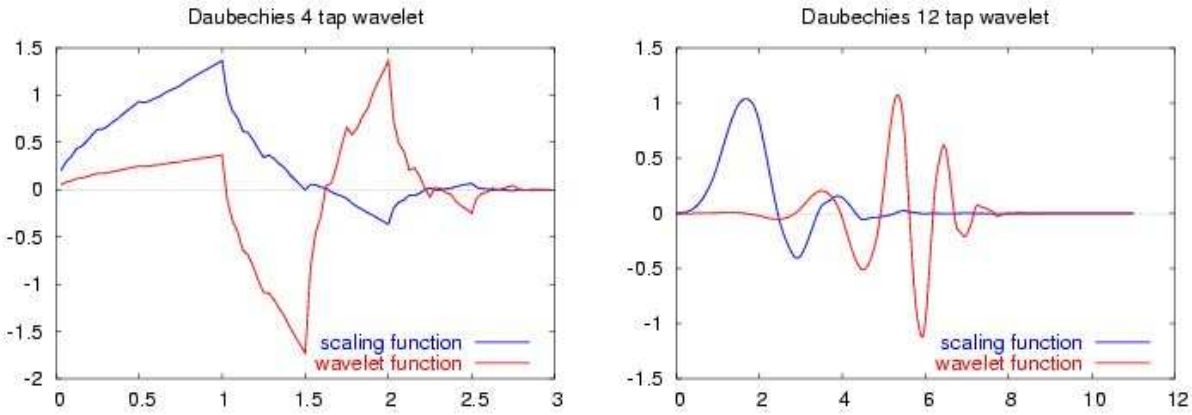
kde je zvolen konstantní faktor $2^{j/2}$, aby uspokojil potřeby normalizace [26].

Daubechiesové vlnky

Daubechiesové vlnky patří do rodiny ortogonálních vlnek a jsou pojmenovány podle jména jejich objevitelkyně, belgické fyzičky a matematičky Ingrid Daubechiesové [23]. Používají se u diskrétní vlnkové transformace a jsou charakteristické maximálním počtem nulových momentů. Pro každý typ vlnky v této třídě je definovaná rozměrová funkce (také zvaná jako *otcovská vlnka*).

Daubechiesové vlnky byly vybrány, aby měly nejvyšší počet *A* nulových momentů (to neznamená nejlepší vyhlazení). Pro danou podporu šířky N=2A je tak mezi 2A-1 možnými řešeními vybráno to, kterého rozměrový filtr má extrémní fázy. Daubechiesové vlnky mají rychlou vlnkovou transformaci a jsou i díky tomu využívaný na řešení široké škály problémů, jako jsou fraktálové problémy nebo, jako j tomu v této práci, k detekci pohybu v obraze.

Tyto vlnky nejsou definovány z hlediska konečné rozměrové a vlnkové (mateční) funkce. Není je totiž možné zapsat v nějaké uzavřené formě. Grafy rozměrové a vlnkové funkce zobrazené na Obrázku 2.14 tak jsou generovány pomocí tzv. *kaskádového algoritmu*[7].



Obrázek 2.14 Průběh rozměrové a vlnkové funkce pro Daubechies typu 4 a 12 [23].

Běžně se používají Daubechiesové ortogonální vlnky typů D2-D20. Číslo zde reprezentuje počet užitých koeficientů *N*. Každá vlnka má počet nulových nebo mizících momentů rovný polovině počtu koeficientů. Například D2 (což je Haarova vlnka) má jeden nulový moment, D4 má dva, atd. Nulový moment pak má tu vlastnost, že limituje schopnost vlnky representovat polynomiální chování nebo informaci v signálu. Například D2 snadno zakóduje 1 polynomiální koeficient.

Další typy vlnek

Mezi další používané vlnky patří *Coifletovy vlnky*, které byly navrženy tak, aby byly více symetrické oproti Daubechiesovým vlnkám [24]. Rozměrové funkce Coilfetových vlnek mají pouze N/3-1 nulových momentů a jejich vlnková funkce má N/3 koeficientů. Ve své práci pak využívám typu C6.

Legendrovy vlnky [21] jsou kompaktně podporované vlnky, které vychází z *Legendrových polynomů*. Legendrovy funkce mají široké uplatnění v aplikacích, kde je používán kulový koordinační systém. Jako u mnoha jiných vlnek, ani zde neexistuje jednodušší analytická formule, která by tyto harmonické kulové vlnky popisovala. Nízko-pásmový filtr, který je asociován s Legendrovými multirozlišovacími analýzami, je filtr s konečnou impulzní odezvou (*FIR*). Tyto vlnky jsou tedy běžně preferovány v mnoha aplikacích. To i přesto, že Legendrovy vlnky nejsou ortogonální.

Legendrovy vlnky $\psi_{n,m}(x)$ jsou popsány následující rovnicí [25]:

$$\psi_{n,m}(x) = \begin{cases} \left(\frac{2m+1}{2}\right)^{\frac{1}{2}} 2^{\frac{k}{2}} p_m\left(2^k x - \hat{n}\right) & \frac{\hat{n}-1}{2^k} \leq x < \frac{\hat{n}+1}{2^k}, \\ 0 & \text{jinak,} \end{cases}$$

kde $k = 1,2,\cdots, \hat{n} = 2n-1, n = 1,2,\cdots 2^{k-1}$ a m je pořadí Legendrova polynomu $p_m(x)$.

3 Popis častých komponent u detekce a sledování pohyblivých objektů

V této části budou popsány jednotlivé komponenty, které bývají v závislosti na způsobu řešení nezbytné k vytvoření *detektoru* a *trackeru* pohybujících se objektů. Samozřejmě je možné, že v závislosti na typu implementace zde některé důležité prvky chybí nebo naopak přebývají. Já se zde chci zaměřit hlavně na ty prvky, kterých dále využívám při implementaci.

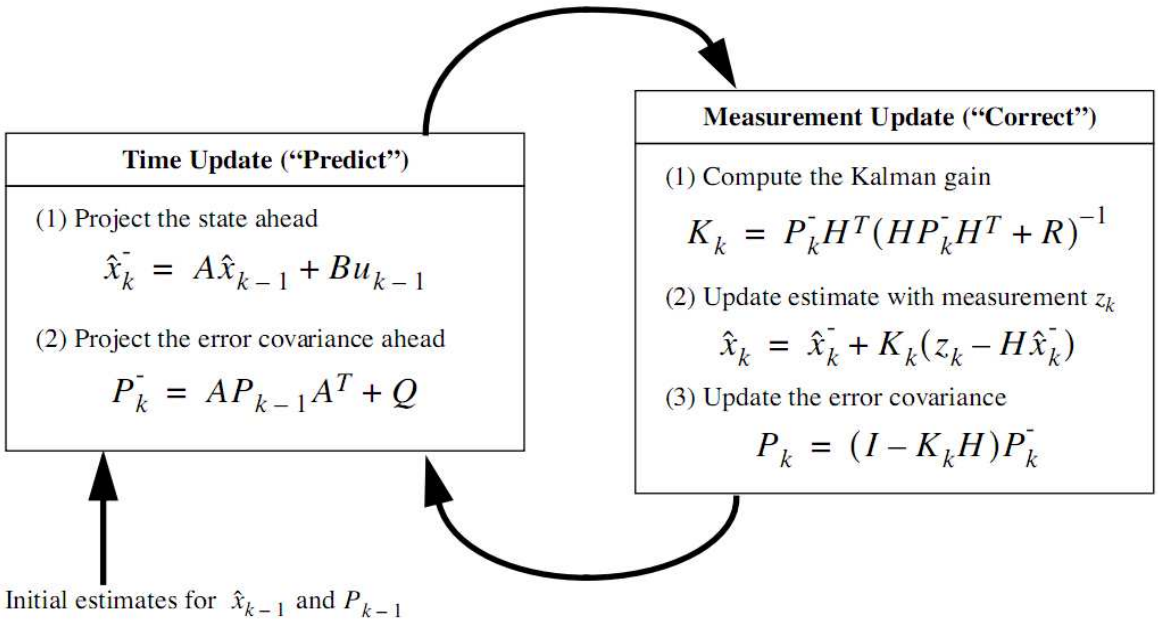
3.1 Kalmanův filtr

V roce 1960 R. E. Kalman publikoval svojí práci popisující rekurzivní řešení, které řeší problém lineárního filtrování diskrétních dat. Od té doby je Kalmanův filtr intenzivně zkoumán, zdokonalován a využíván.

Jde o sestavu několika matematických rovnic vycházejících z Bayesovské filtrace, které zajišťují efektivní výpočet, který určuje stav zkoumaného procesu, tak aby byla minimalizována chyba. Filtr podporuje odhad minulých, současných a budoucích stavu zkoumaného procesu. A to i v případě kdy je nám přesná podoba modelovaného systému neznámá.

Algoritmus

Kalmanův filtr odhaduje stav procesu pomocí zpětné kontroly. Filtr odhadne stav v daný čas a potom



Obrázek 3.1 Znázorňuje diagram, který popisuje, jak Kalmanův filtr funguje společně s jeho rovnicemi. [9]

zajistí kontrolu odhadu pomocí měření skutečného stavu, které může obsahovat nepřesnosti (šum). Kalmanův filtr tedy pracuje ve dvou základních stavech, které jsou popsány vždy několika rovnicemi v Obrázku 3.1.

Fáze „Time Update“ (z Obrázku 3.1) je zodpovědná za předpověď následujícího stavu. A update nové chyby měření. To zajistí první odhad pro další krok. Druhá fáze „Measurement Update“ je fáze opravy predikce na základě měření. Je zde aktualizován takzvaný *Gain* Kalmanova filtru a dále odhad stavu a chyba měření.

A, které se vyskytuje v rovnici (Obrázek 3.1), je translační matice a určuje vztah mezi předchozím krokem *k-1* a současným krokem *k*. Jedná se o matici rozměru $n \times n$, přičemž v praxi se hodnota matice může v každém kroku měnit, ale obecně se předpokládá, že je konstantní. Matice *B* o rozměrech $n \times l$, je kontrolní matice, která se volitelně používá, je-li na vstupu $u \in \Re^l$ k danému stavu *x*. *H* je matice o rozměrech $n \times m$ a jde o matici měření, která má vztah k naměřené hodnotě $\widehat{X_k}$. Stejně jako matice *A* i *H* se může každý krok měnit. Dále jsou v rovnicích obsaženy tři šumy a Kalmanův *gain* (zisk). *R* je šum, který vzniká při měření vlivem nepřesnosti měření. *Q* je šumem vznikajícím během výpočtu rovnice vlivem různých zaokrouhlovacích chyb. *P* je pak odhad chyby výpočtu. Přičemž *Q* a *R* mohou být na počátku konstanty a *P* společné z *K* se po úvodní inicializaci také rychle stabilizují a pak už mají konstantní průběh. Viz [9].

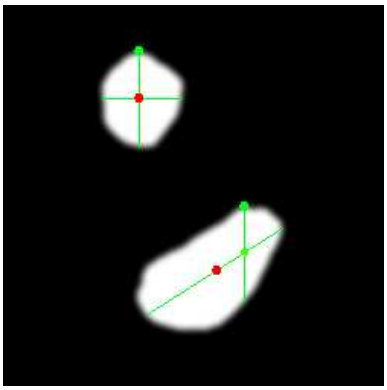
3.2 Metody detekce blobů

V počítačovém vidění jde o metodu, která detekuje body a regiony v obraze, které jsou světlejší nebo tmavší než okolí. V této práci se zabývám detekcí blobů, protože hledám zajímavé regiony pro budoucí zpracování (sledování). Vyhledávání blobů má i mnoho jiných významů jako je vyhledávání části obrazu z důvodů jejich rozpoznání nebo segmentace obrazu. Dále uvedu některé zajímavé metody používané k detekci blobů. Nejvíce mě zaujaly metody od Erika van Kempena uvedené v [10]. Jde jak o vlastní metody, tak i o převzaté.

Přehled jednoduchých metod

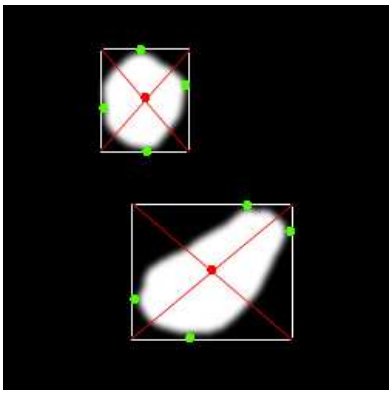
Jak jsem již psal výše, v článku [28] jsou uvedeny 4 autorovy návrhy jak detekovat bloby. Jde o pomalejší a výpočetně náročnější algoritmy.

První metoda je nazvána $Line_{max}$. Myšlenkou je postupným procházením pixelů obrazu najít první bílý pixel, který značí vstup do blobu. Z tohoto bodu se pak vede vertikální přímka až k výstupnímu bodu z blobu. U vzniklé vertikální úsečky se pak určí střed a hledá nejdelší úsečka v blobu, která prochází středem vertikální úsečky. Pak se zjistí střed této nejdelší úsečky a tento střed je definován jako středový bod blobu (Obrázek 3.2). Problém algoritmu je, že příliš malé bloby velikosti jednoho pixelu nezaznamená. Rychlost je pak závislá na počtu iterací při vyhledávání nejdelší přímky v blobu. A tedy, čím větší blob, tím pomalejší při jeho hledání.



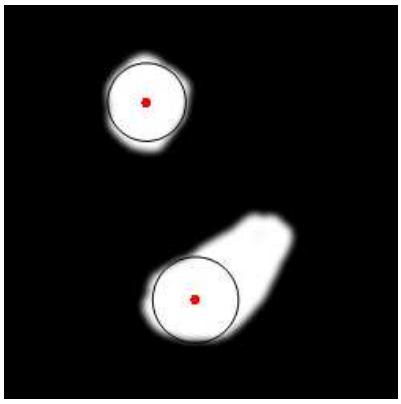
Obrázek 3.2 Algoritmus $Line_{max}$ [28].

Metoda *Vnějšího obdélníků* hledá největší a nejmenší hodnoty daného pixelu na osách *x* a *y*. Podle toho se pak určí rohy obdélníků a uprostřed tohoto obdélníku se podle jeho diagonál určí střed blobu (Obrázek 3.3). Problémem tohoto přístupu je, že není jednoduché najít minimální a maximální hodnoty blobu na osách *x* a *y*. Dalším problémem je, že metoda má zvláštní chování u specifických tvarů blobů jako je třeba půlměsíc (střed blobu může být mimo blob). Rychlost metody je pak lineární vzhledem k velikosti blobu.



Obrázek 3.3 Algoritmus Vnějších obdélníků [28].

U *Vnitřní kružnice* je cílem najít co největší vnitřní kružnici pro každý pixel v blobu. Střed kružnice se pak definuje jako střed blobu (Obrázek 3.4). Problémem je, že je potřeba stále kontrolovat zda každý nový bod, který se zkouší je v tom samém blobu jako ostatní body. Tento algoritmus je velmi pomalý, protože se jeho výpočetní náročnost zvyšuje exponenciálně společně s velikostí blobu.



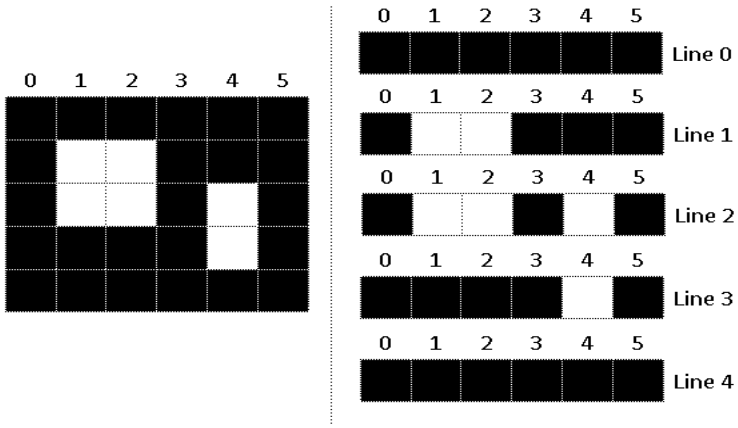
Obrázek 3.4 Algoritmus Vnitřní kružnice [28].

Poslední zmiňovanou metodou je *Vnitřní obdélník*. Tato metoda funguje na stejném principu jako předchozí *Vnitřní kružnice*. Je stejně pomalá a problémová.

Rostoucí regiony

Rostoucí regiony jsou velmi jednoduchým a rychlý algoritmem vhodným zejména pro dvouhodnotový (černobílý) obraz. Tento algoritmus nyní využívám v mé aplikaci.

Obraz, který je reprezentován maticí, je postupně po řadách procházen. Když je v řadě nalezena skupina jednoho či více odlišných pixelů, tak je z nich utvořen tzv. *lineblob*. Postupně se tak procházejí všechny řady obrazu a z nich se sbírají *lineblobs*. Potom se znovu prochází jednotlivé *lineblobs* a hledá se, zdali se na dvou vedlejších řádcích překrývají, pokud ano, tak se spojí do jednoho většího blobu. Viz [27] a Obrázek 3.5.



Obrázek 3.5 Grafické znázornění algoritmu rostoucích regionů [27].

Laplacian of Gaussian (LoG)

Laplacian of Gaussian je jedním z prvních a nejběžnějších detektorů. Na vstupu je obrázek jako funkce $f(x, y)$, ten je následně prohnán konvolucí z *Gaussian* jádrem

$$g(x, y, t) = \frac{1}{2\pi t} e^{-(x^2+y^2)/(2t)}$$

pro určitý rozsah *t* dostaneme rozsahově prostorovou reprezentaci

$$L(x, y, t) = g(x, y, t) * f(x, y).$$

Potom je Laplacian operátor spočítán jako

$$\nabla^2 L = L_{xx} + L_{yy}.$$

To má většinou za následek silně pozitivní odezvu pro tmavé bloby a silně negativní odezvu pro světlé bloby podobné velikosti. Problémem zde je, že odpověď *Laplacian* operátoru je závislá na vztahu mezi velikostí blobu v obraze a velikostí *Gaussian* jádra, jenž je použito. Proto se pro rozdílně velké bloby používá multi-průchodový přístup. Výše zmíněný přístup, ale i jiné lze najít na [11].

3.3 Proces sledování objektů

Sledování objektů může být velmi náročný problém, obzvlášťě když se objekt nepohybuje po lineární dráze a jeho pohyb je v zásadě zcela náhodný. Dále bývá problémem, když objekt na okamžik zmizí a pak se opět objeví na jiném místě. Zde může být problém rozpoznat, že jde o ten samý objekt. Ten samý problém s přiřazením správného objektu nastává při sledování více objektů, které se míjejí.

Při sledování objektu bychom tedy měli o něm vědět co nejvíce informací, abychom náš sledovaný objekt v každém dalším snímku bezpečně rozeznali. Existují dva základní přístupy pro vizuální sledovací systémy [12]. *Target Representation and Localization* a *Filtering and Data Association.*

Target Representation and Localization jsou ze spodu nahoru jdoucí metody. Typicky je výpočetní komplexnost těchto algoritmů velmi nízká. Některé běžné algoritmy využívající výše zmíněný přístup jsou *Blob tracking* (založeno na segmentaci vnitřku objektů), *Kernel-based tracking (Mean-shift tracking),* který je iterativní lokalizační procedurou založené na podobnosti měření. Dále to mohou být *Contour tracking* (detekce hranic objektů) a *Visual feature matching.*

Filtering and Data Association jsou většinou metody jdoucí ze shora dolů, které zahrnují základní informace o scéně či objektu. Dovedou se vypořádat s dynamikou objektu i s vyhodnocováním různých hypotéz. Výpočetní komplexnost těchto algoritmů je obecně velmi vysoká. Patří mezi ně již popsaný *Kalmanův filtr* nebo *Částicový filtr,* který je vhodný pro vzorkování základního stavového prostoru nelineárních a *non-Gaussian* procesů. Viz. [12].

3.4 Metody odečítání pozadí

Cílem metod založených na odečtu pozadí je určit, které části obrazu se mění a kde se tedy vyskytuje pohyb. Obraz je těmito metodami rozdělen na statické pozadí (*background*) a pohybující se popředí (*foreground*), které může být dále zpracováváno.

Naivním způsobem řešením pak může být porovnávání aktuálního snímku vzhledem k nějakému statickému vzoru. Problém však je, že by se řešení mělo být schopno adaptovat na případné změny osvětlení a stíny. Eliminovat některé stále se opakující vysokofrekvenční pohyby jako jsou oscilace kamery nebo pohyb listí či déšť ve scéně.

Tato problematika velmi často zkoumána a byla tedy vyvinuta široká škála algoritmů. Některé z nich zde budou představeny.

Gaussian Mixture modely

U metody *Gaussian Mixture* [13] je každý pixel obrazu modelován zvlášť jako směs *k* Gaussovských funkcí. Cílem je, aby byl pixel pokryt co největším počtem Gaussovkých funkcí (rovnice 3.1).

Takový pixel pak může být označen za pozadí. A naopak pixely, u kterých není žádná shoda, jsou považovány za popředí.

$$\boldsymbol{P}(\boldsymbol{I_t}) = \sum_{i=1}^{k} w_{i,t}\eta(I_t; \mu_i; \Sigma_{i,t}), \tag{3.1}$$

kde I_t je barva pixelu (*x*, *y*) v obraze *I*, $\mu_{i,t}$ je průměrový vektor, $\Sigma_{i,t}$ je kovariantní matice *i*-té Gaussovky v čase *t* a w_i je váha asociovaná u každé gaussovky.

Odečet pozadí na základě barvy

U odečtu pozadí na základě barvy se využívá změn v barevnosti obrazu. Existuje několik metod, které pracují na tomto principu, jako jsou algoritmy *Improved Hue, Luminance and Saturation* a *YUV Colour Space*.

Improved Hue, Luminance and Saturation algoritmus [13] počítá u zkoumaného pixelu jeho rozdíl mezi průměrnou barvou jeho vektoru a pozadím obrázku. Jako popředí klasifikuje ty pixely, u kterých je výrazný rozdíl barevností a jasové hodnoty.

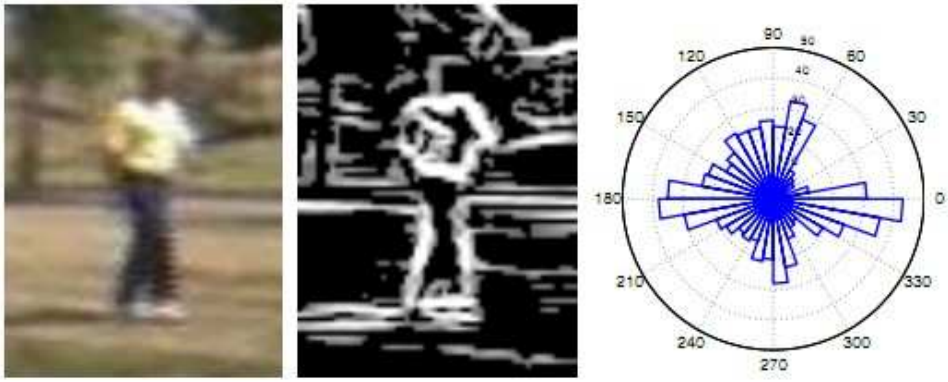
YUV Colour Space algoritmus [13] je založen na výpočtu prahu na základě měření různých rozdílu mezi zkoumaným pixelem a pozadím. Pixel je pak klasifikován jako popředí pokud platí, že $d(i,j) - d$ je větší než vypočítaný práh. *d* je zde průměrná hodnota rozdílu obrazu.

4 Používaná řešení

Problematika detekce a sledování objektů ve statickém obraze není nová. Přede mnou už bylo navrženo mnoho rozličných systému řešení. Zde bych rád čtenáře seznámil s jiným přístupem.

Sledování více objektů pomocí hierarchického částicového filtru

Tato metoda byla publikována v [34]. Jde o velmi robustní algoritmus založený na částicovém filtru. Sledovaný objekt zde je charakterizován pomocí histogramu barev a hranově orientovaného histogramu (Obrázek 4.1). Více vzorků a rysů (jako jsou zmíněné histogramy) pak sice může zvýšit robustnost, na druhou stranu se neúměrně zvyšuje výpočetní náročnost částicového filtru. Proto jsou zde v rámci zrychlení algoritmu a zachování robustnosti jistá vylepšení. Prvním z nich je užití integrálních obrázků. Ty mají za úkol efektivněji počítat barevné příznaky a hranově orientované histogramy, což dovoluje použít větší počet částic a lepší popis cílů. Sledovaná pravděpodobnost, která je založená na více rysech, je počítána Coarse-to-fine [35] algoritmem, který dovoluje se výpočetně rychle zaměřit na nejzajímavější regiony obrazu. Kvazi náhodné vzorkování částic pak dovoluje filtru dosáhnout vyššího convergetního stupně. Výsledný algoritmus pak je odolný vůči šumu v obraze či krátkodobé okluze.



Obrázek 4.1 Hranově orientovaný histogram. Vlevo je vzorový obrázek, uprostřed hranově zesílený obrázek a vpravo je polární graf hranově orientovaného histogramu [34].

5 Návrh řešení

Na základě teoretických poznatků, které jsem prezentoval v předchozích kapitolách, navrhuji v této části práce systém, který by měl být schopen detekovat a sledovat vybranou třídu pohybujících se objektů ve statickém obraze. Navrhované řešení vychází z první verze řešení, které jsem vytvářel v rámci semestrálního projektu. Základní struktura by měla zůstat stejná. Jedná se o obecně používanou strukturu pro tento typ aplikací.

5.1 Zaměření navrhované aplikace

Nejprve je třeba si definovat cíle aplikace. Tedy typy scén, které by měl být systém schopen řešit. Již ze zadání vyplývá, že bych měl být schopen detekovat a sledovat malé pohybující se objekty. Objekty velikosti v řádech pixelů. Podle toho je pak třeba stanovit vhodné metody, které se použijí.

První verze programu v semestrální části projektu byla dělaná na podmínky interiéru. Tam byl problém pouze se šumem obrazu v důsledku nedostatečného osvětlení. Konečná verze by si ale měla být schopná vyrovnat s venkovním prostředím a tedy i vlivy, které takové prostředí přináší. To může být náhlá změna sluneční intenzity nebo nejrůznější vysokofrekvenční periodické pohyby jako může být déšť nebo pohyb větví stromu. Mojí snahou při získávání sady testovacích videí v terénu, tedy mimo jiné bude se co nejvíce vyvarovat nejhorším rušivým podmínkám. Vyloučit je však zcela nelze a navrhovaná aplikace si musí poradit se slabšími rušivými vlivy, které venkovní prostředí přináší.

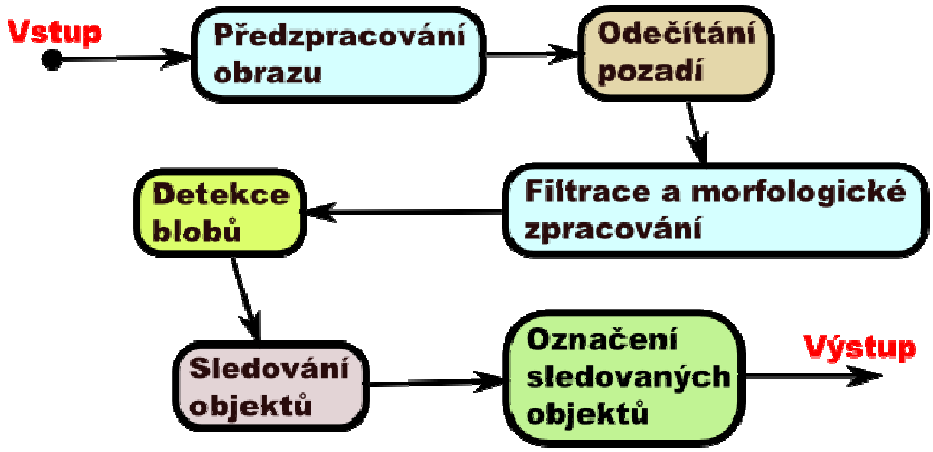
Za cíl této práce stanovuji, že by měl být vyvinut systém, který by měl být schopen detekce a sledování aut jedoucích ve větší vzdálenosti. To je alespoň 200-300 metrů daleko. U tohoto typu sledování se nevyhnu základním problémům, jako je pohybující se *zeleň* (tráva, stromy, keře) v obraze. Dalším problémem muže být skutečnost, že v České republice jsou silnice hojně lemovány stromy a stromky. Z toho vyplývá řešení situace, kdy se sledované vozidlo ztrácí a znovu objevuje za stromem. Na druhou stranu se zde vzhledem k velikosti objektů nemusím zabývat stínem, protože ten v tomto případě splývá s pohybujícím se objektem.

Aplikací bych chtěl dále využívat i v interiéru a detekovat tak malé míčky, autíčka, nebo jak zkouším v jedné sadě testovacích videí, sledovat malý model helikoptéry vznášející se po místnosti. To je totiž ta správná ukázka zcela náhodného pohybu a test navrhovaného *sledovače*.

5.2 Návrh celkové struktury aplikace

Vyjdu-li z jiných příkladů zabývajících se daným problémem ([15] nebo [29]], tak každá aplikace daného typu musí mít v prvé řadě nějaký detektor a sledovač objektů. To je velmi zjednodušený pohled. Těch jednotlivých částí je více. Nejdříve je třeba načíst a předzpracovat obraz, pak na obraz

aplikovat jednu z metod odečítání pozadí, kterých mám pro porovnání více. Na výstup z odečítání pozadí se provedou další dodatečné operace filtrace, aby se odstranil šum, a výstup se užije k detekci blobů. Tím se získají pohybující se objekty, které se dále sledují. Sledované objekty se na závěr v obraze označí, například zeleným obdélníkem. To jde pak na výstup aplikace. Celé je to znázorněno na Obrázku 5.1.



Obrázek 5.1 Navrhovaná struktura algoritmu detekce a sledování objektu

Uživatelské rozhraní

Co se týče uživatelského rozhraní, tak by mělo mít konzolový charakter, kdy se všechny nastavení programu zadávají při spouštění programu. Po jeho spuštění by pak měl být vidět real-timový průběh. Mělo by se zadávat co nejméně parametrů. Snahou je to omezit na parametry určující jména vstupního, výstupního souboru. Dále pak typ použité metody odečítání pozadí a typ použitého testu, provádí-li se. Po experimentech bych tam přidal ještě nastavení týkající se sledování objektu.

5.3 Návrh metod odečítání pozadí

První fází algoritmu je tedy odečítání pozadí. Jde o klíčovou část projektu, a proto zde chci porovnávat více metod. Už první verze projektu naznačila, že musí jít o dostatečně rychlou a přesnou metodu, která se dokáže vypořádat se středně velkým množstvím šumu.

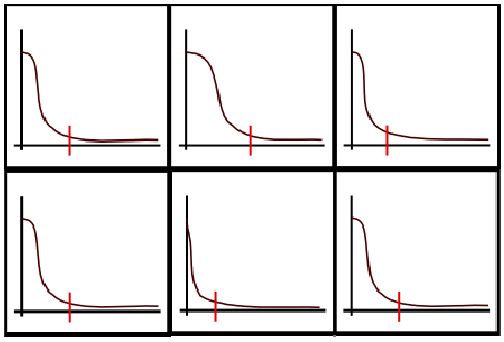
Na základě práce popsané v [15] jsem se rozhodl zde využít k odečítání pozadí *vlnek*. V [15] je popsané odečítání pozadí pomocí *Daubechiesových vlnek*. Já to pro srovnání chci aplikovat i na další vlnky. Vybral jsem si *Coifletovy vlnky* a *Legendrovy vlnky*. Všechny tyto vlnky už jsem popsal v kapitole 2.6, kde se vlnkám podrobněji věnuji. Pro porovnání výsledků pak chci také detekovat pozadí pomocí prostého porovnávání posledních dvou obrázků videa a pomocí algoritmu *Mixture of Gaussian*.

Má metoda, stejně jako v [15], používá *Daubechiesové vlnkový filtr* o délce 4, 6 a 8, tedy D4, D6, D8, jak je popsáno v [23]. Tyto filtr se pak aplikují na sekvence šedotónových obrázků o délce 4, 6 a 8 podlé typu filtru. Vysoko pásmová vlastnost filtru by pak měla zvýraznit siluetu pohybujícího se

objektu, jak je ukázáno na Obrázku 5.2. Výhodou toho přístupu by měla být robustnost a odolnost vůči šumu. Výsledek, který vzejde z *Daubechiesového vlnkového filtru* je pak dále filtrován. Nejprve se pomocí vyhlazení vhodným konvolučním jádrem odfiltrují vysoké špičky energie v obraze. Potom se prahováním obraz převede na binární obraz, který je dále upraven pomocí morfologických operací.

Obrázek 5.2 Výstup D4 filtru, ukazuje zachycení pohybu hranic objektu [15].

Mým vylepšením algoritmu pak je u prahování, kde se snažím práh nastavovat automaticky lokálně a každý snímek znovu. Toto řešení má za úkol potlačit lokální šum, který vzniká vlivem nežádoucího pohybu *zeleně* v obraze a také potlačit náhlé změny osvětlení. Mělo by to fungovat tak, že se obraz rozdělí na malé oblasti o velikosti hrany několika desítek pixelů. V každé takové oblasti se pak vytvoří histogram jasu, který má gaussovský průběh. V histogramu se pak ve směru zvyšujícího se jasu určí zlom klesání gaussova průběh a v tom místě se stanoví práh pro danou oblast. To je načrtnuto na Obrázku 5.3.



Obrázek 5.3 Náčrt metody lokálního určování prahu v obraze.

Co se týče morfologický operací, které upravují obraz, tak navrhuji operaci binárního otevření následovanou znásobením binárního uzavření. A navrch ještě provést dilataci s menším strukturním elementem. To za účelem zvětšení daného blobu. Přitom by se nijak zvlášť neměla měnit skutečná velikost detekovaného objektu. Stejně je totiž často zmenšen už během předchozích operací, mělo by jít o jistou korekci. Pokud jde o termín znásobeného binárního uzavření. Tak jde o několika násobnou dilataci následovanou několika násobnou erozí. Cílem je zaplnit větší díry a zálivy v objektech a získat tak místo několika menších objektů jeden větší.

Jak jsem psal výše, pro širší porovnání metod jsem se rozhodl využít u výše popsané metody i jiných vlnek. A to *Coifletovy* a *Legendrovy* vlnky. V celém algoritmu by se nemělo nic měnit, akorát

by se měli zaměnit hodnoty používaných nízko a vysoko pásmových koeficientu h_n a g_n, tedy koeficientu měřítka (*scale*) a vlnky.

Dalšími metodami pro porovnání by mělo být prosté porovnání snímku a změny v něm. To bych dělal kvůli zjednodušení pro šedotónový obraz. Tento postup by měl vykazovat vysokou hodnotu náhodného šumu. Zde stačí práh nastavit napevno. Druhou slabší porovnávací metodou je *Mixture of Gaussian*, který je již implementován v balíku knihoven OpenCV, které hodlám pro tento projekt využít. Tuto metodu jsem navíc již použil v první verzi programu v rámci semestrálního projektu.

5.4 Detekce blobů

Zde hodlám využít metodu *Rostoucích regionů* popsanou v [27]. Už v semestrální verzi se tato metoda osvědčila. Byla blíže popsána v podkapitole 3.3. Jedinou její omezující podmínkou je, že přijímá pouze binární obraz.

5.5 Sledování objektu

Pro sledování objektu chci použít Kalmanův filtr, tuto metodu jsem již používal, je přesná a je používaná v mnoha jiných pracích podobného zaměření [15][29]. Výstupem z Kalmanova filtru by pak měly být souřadnice polohy objetu v daný moment. Podle toho se pak vykreslí do okna, kde běží vyšetřované video, obdélníček kolem pohybujícího se objektu.

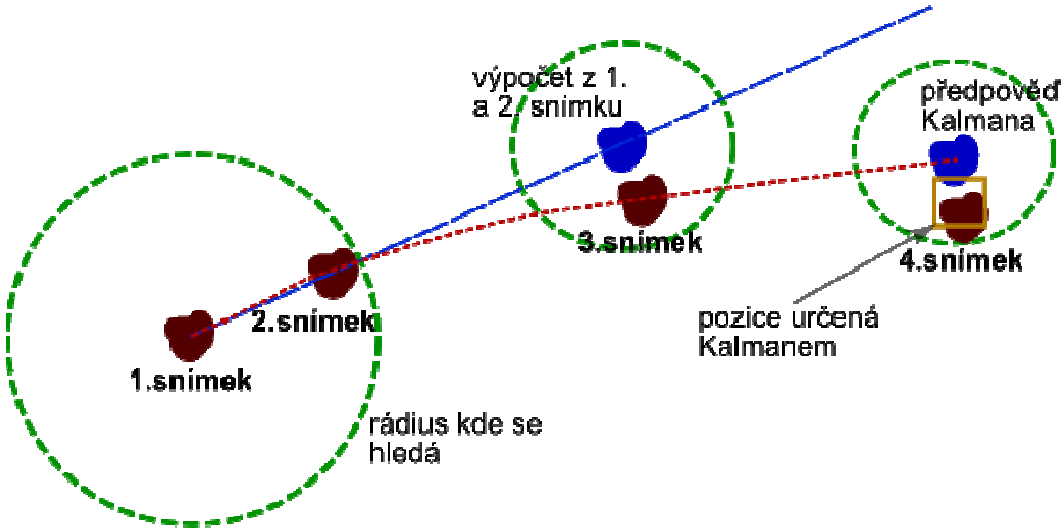
Tato část systému je druhou nejdůležitější částí, protože chceme-li měřit přesně polohu jakéhokoliv pohybujícího se objektu, tak je nezbytné jí i co nejpřesněji měřit a sledovat. K tomu právě sloužit správná inicializace Kalmanova filtru.

Prvním problémem je kdy započít samotné sledování pohybujících se objektů. Já jsem se rozhodl podle vzoru v [15] začít poté, když jsou nalezeny nějaké bloby alespoň ve třech za sebou jdoucích snímcích. V tu chvíli by mělo začít vyhledávání potenciálních objektů ke sledování následované inicializací Kalmanova filtru u pohybujících se objektů. Od 4. snímku pak jejich sledování pomocí Kalmanova filtru.

Ztratili se a znovu objeví nějaký objekt, tak by měl být můj systém schopen překonat díru jednoho snímku bez objektu. Případně i více, maximálně asi tak 2-3, protože potom se při úplném zmizení objektu začíná sledování dopouštět záměrné a nežádoucí chyby. Objeví-li se v průběhu sledování úplně nový blob, tak jsou potřeba vždy první 3 snímky pro novou inicializaci, která je následující.

Za prvé se vezme blob z 1. snímku a v daném okolí se hledají na druhém 2. snímku bloby v dosahu. Za druhé mezi blobem na 1. snímku a blobem na 2. snímku se vypočte rychlost a její směr a předpoví se, kde bude sledovaný blob ve 3. snímku. A za třetí se ve 3. Snímků na předpovězeném

místě a daném okolí hledá další blob. Je-li nalezen, tak se může inicializovat Kalmanův filtr pro sledování daného blobu a od 4. snímku být sledován a označen. Popsaný algoritmus je znázorněn na Obrázku 5.4.



Obrázek 5.4 Ilustrace algoritmu inicializujícího Kalmanův filtr.

Kalmanův filtr pak tedy může být inicializován a předpovídat pozici budoucího blobu a v dané oblasti pak blob hledat. Hodnota, která je pak naměřena v reálu se opět předá Kalmanovu filtr, který připočte chybu měření a určí polohu blobu. Není-li blob nalezen tam, kde Kalman očekává, tak je sledování daného blobu přeskočeno a Kalman se blob snaží najít na očekávaném místě v dalším snímku. V případě, že ani na podruhé neuspěje, tak je sledování daného blobu (objektu) ukončeno. Jak bylo popsáno výše, může se přeskočit i více snímku.

Inicializace Kalmanova filtru

Základem inicializace filtru je nastavení inicializačních hodnot matic a šumů, které se po celou dobu nemění a nabývají konstantních hodnoty. Pokud jde o počáteční hodnoty Kalmanova *gain*u a odhadu chyby výpočtu, tak na jejich inicializaci až tak moc nezáleží, protože se časem stabilizují na konstantní hodnotě.

Nejdůležitější jsou pak inicializace translační matice A a měřící matice H, které se obecně stanovují jako

,

kde A je translační nebo také dynamická matice a určuje vztah mezi k-tým a $k-1$ stavem systému. Matice je stejně velká jako počet atributů, které každý stav obsahuje (například pozici, velikost a rychlost ve 2D). Dále pak

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix},$$

kde H je pak měřící matice a určuje vztah mezi nově naměřenou hodnotou a její korekturou. Počet řádků matice udává počet atributů, které se u stavu nově měří a šířka potom počet atributů, kterými je stav systému popsán.

Na závěr se musí na základě typu měření a zpracování dat v rovnicích Kalmanova filtru určit šumy procesu a měření. Šum procesu by se měl nastavit na nějakou obecně známou odzkoušenou hodnotu pro danou implementaci Kalmanova filtru. A pro šum měření použít nějakou doporučenou hodnotu či testovat různé varianty.

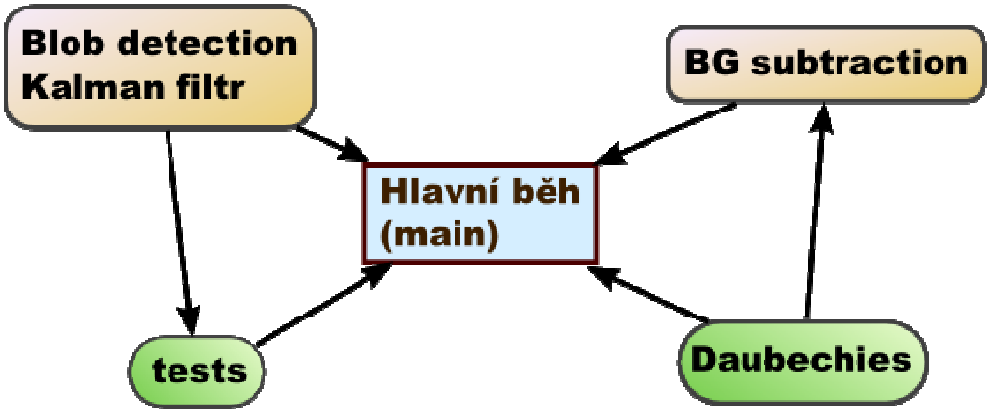
6 Implementace

V rámci semestrálního projektu jsem vytvořil první verzi programu do mé diplomové práce. Na této verzi jsem dále stavěl a rozvíjel ji. Projekt jsem řešil v jazyce C++ ve vývojovém prostředí MS Visual C++ 2008 s využitím knihovny OpenCV.

OpenCV znamená Open Source Computer Vision Library a jde o kolekci asi 300 C funkcí a několika C++ tříd, které implementují často používané algoritmy zpracování obrazu a počítačového vidění. Jde o multiplatformní rozhraní pro programování, které je použitelné i v programovacím jazyce Python.

6.1 Struktura aplikace

Základní kostra programu je odvozená od vzorových programů pro práci z videi v OpenCV. Aplikace je pak rozdělena na 5 hlavních částí, z kterých jsou dvě třídy. Jde o hlavní běh, kde se zpracovávají parametry, otevírá se zde a přehrává vstupní video a volají odtud všechny návazné funkce. Pak tu je knihovna seskupující funkce odečítání pozadí, které jsem implementoval, a filtry upravující výstup odečtu pozadí. Na ní úzce navazuje třída *Daubechies*, která zastřešuje všechny vlnkové filtry. Pak tu je knihovna obsahující struktury a funkce potřebné k detekci blobů a k sledování Kalmanovým filtrem. Poslední prvkem pak je třída *tests*, která slouží k testování a její metody budou popsány v 7 kapitole. Celá struktura je znázorněna na Obrázku 6.1.



Obrázek 6.1 Struktura aplikace, směr šipky značí, kde se daný modul používá.

6.2 Hlavní běh

Jak již bylo napsáno výše, *main* je řídící jádro aplikace. Jde o jednu jednolitou funkci. Nejprve se zde zpracovávají vstupní parametry a podle nich se nastavují řídící proměnné. Tato část je řešená pomocí sekvence podmínek tak, aby zde bylo podpora proměnného počtu parametrů. Některé parametry jsou povinné, ostatní volitelné.

Po nastavení parametru se otevírá video, které by mělo být kódované pomocí kodeku DivX s frekvencí 15snímků za 1 sekundu. Funkce, které jsou z OpenCV a starají se o otevírání a streamování videa by měly zvládat i jiné vstupy. Jde ale o to, že do kodeku DivX s frekvencí 15 snímku za 1 sekundu se pak ukládá výsledné video.

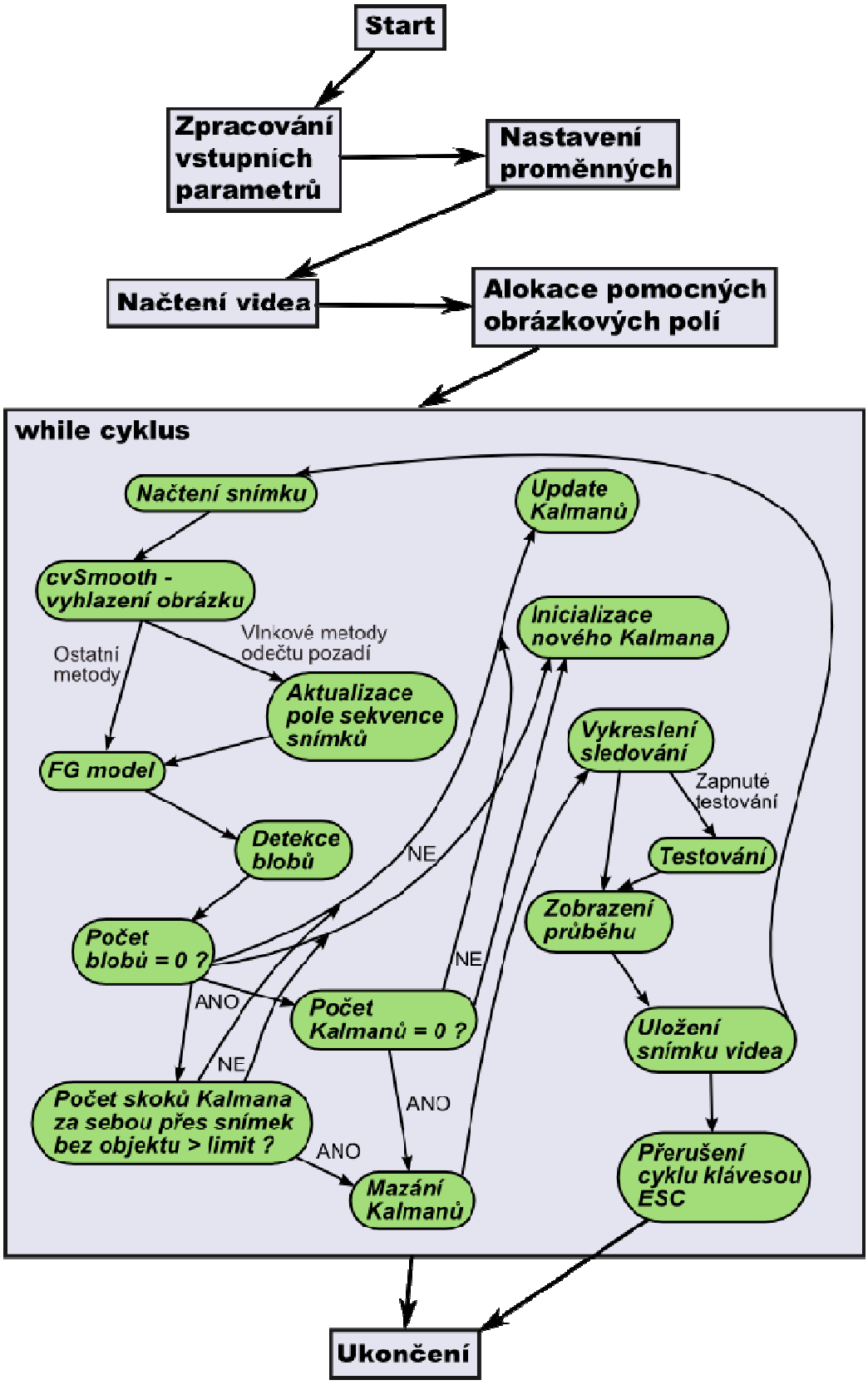
Po té co je známo rozlišení vstupního videa, tak se alokují pole pro uchovávání pomocných obrázků. Vlnkové metody odečítání pozadí potřebují například uchovávat historii 4 až 8 posledních snímků. Dále je zde potřeba alokovat pole pro snímek kde bude výsledek odečítání pozadí vlnkou a pak také *foreground* model, který znázorňuje pohybující se bloby.

Pak nastupuje hlavní funkční část. Jde o cyklus, kde se vždy na začátku načítá 1 snímek z videa, který je dále zpracováván. Nad každým snímkem jsou provedeny v *pseudo pipeline* následující operace. Nejprve se obrázek vyhladí pomocí funkce `cvSmooth` z OpenCV. Dále v případě, že jde o odečet pozadí pomocí diferenční metody nebo *Mixture of Gaussian*, tak se rovnou získá model pohybujícího se popředí (foreground model = FG model). U vlnkových metod se nejprve aktualizuje pole snímků z pamětí posledních 4-8 snímku. Pak se i zde podle typu metody získá FG model.

Další část cyklu obsahuje sekvenci podmínek, která se stará o správné započetí inicializace sledování a průběh sledování objektů. Důležitý je zde fakt, že pro každou inicializaci sledování nového objektu jsou třeba 3 výskyty blobu v posledních třech snímcích. Nejprve jsou detekovány bloby, které jsou ukládaný do pole vectorů blobů velikosti 3 – 3 poslední snímky s bloby. Pak první `if` kontroluje velikost aktuálního vektoru blobů a také velikost vektoru s Kalmanovými filtry. Jsou-li nalezeny nějaké bloby, tak začíná odpočet detekce. V opačném případě, když nejsou žádné bloby a není ani jeden Kalmanův filtr (nic se nesleduje), tak dojde k vynulování odpočítávání prvních tří snímků. Další `if` řeší sledování objektů pomocí Kalmanova filtru. Každý sledovaný objekt má svůj Kalmanův filtr, který je potřeba v každém snímku aktualizovat, případně smazat, zmizí-li daný objekt. V dalším `if`u se na základě nalezení vhodných blobů v posledních třech snímcích inicializují další sledování pomocí Kalmanových filtrů, případně se mažou neaktivní sledování. Co se týče inicializace Kalmanova filtru, tak matice *A* a *H* popsané v podkapitole 5.5 se nastavují na stejné hodnoty jako v návrhu. Ostatní proměnné (šumy), které se na začátku u každého Kalmanova filtru nastavují, jsem pak převzal ze vzoru popsaného v dokumentaci OpenCV.

Výsledek ze sledování je pak vykreslován obdélníkem do originálního vstupu na patřičné pozici. Tento výsledek se také v případě testování používá na testování. Průběh sledování je na konci cyklu zobrazován ve 2 či 3 oknech, v závislosti na užité metodě odečítání pozadí. Dvě okna se používají pro metodu diference dvou snímků a Mixture of Gaussian. Zobrazuje se vstupní video s označením pohybujících se objektů a FG model. V případě užití vlnkové metody se ještě zobrazuje výsledek vlnkové filtrace, která je mezikrokem při získávání konečného FG modelu. Na úplném konci cyklu se pak původní snímek s označenými pohyby ukládá do výstupního video souboru. Zde je právě použit kodek DivX a s frekvencí 15 snímku za sekundu.

Cyklus pak ještě obsahuje čekání na událost s tisknutí klávesy `Esc`, to pro náhlé ukončení programu. Za cyklem už je pak jen regulérní ukončování aplikace. Celé to pak shrnuje Obrázek 6.2.



Obrázek 6.2 Grafické znázornění běhu řídící funkce main.

6.3 Detekce blobů a sledování Kalmanem

Tato knihovna obsahuje potřebnou podporu pro detekci blobů, sledování blobů (objektů) pomocí kalmanova filtru a struktury, které byly pro tento účel navrženy. Přesně jde o 4 funkce a 6 struktur. Pět struktur má co dočinění s bloby a jedna struktura obsahující Kalman, byla dodána kvůli možnosti přeskakovat snímky, ve kterých se objekt krátce schoval.

Funkce `detectBlobs`, která detekuje bloby z FG modelu, byla převzata z [27] a dále poupravena pro mé prostředí. Na výstupu je vektor, v kterém jsou uložené všechny nalezené bloby. U každého blobu jsou známé jeho rozměry a střed. Klíčovým vylepšením, hlavně pro fázi ladění metod odečítání pozadí, je omezení nalezených blobů na 100 kusů. Jde o to, že v případě špatné metody odečítání pozadí mohlo dojít k tomu, že by se detekovalo příliš mnoho blobů (v řádu tisíců) a došla by paměť počítače.

Ve funkci `getBlobsForKalman` je aplikován algoritmus navržený v podkapitole 5.5 a zobrazený na Obrázku 5.4. Výsledkem pak je struktura, která kromě blobu obsahuje jeho rychlost. Tato struktura je použita v další funkci `initKalman`.

Funkce `initKalman` a `getTrackingBlobsAndKalmanUpdate` jsou odvozeny od vzorového samplu, který obsahuje dokumentace OpenCV. Samotné funkce pak využívají funkcí a struktur, které jsou v OpenCV pro Kalmanův filtr navrženy. Druhá zmiňovaná funkce, která řeší update Kalmanova filtru je pak ještě vylepšená o několik prvků. Za prvé se zde při hledání blobu, který náleži danému Kalmanovu filtru poměřuje předpokládáná velikost nastavená na pevno na 20% – 200% predektivované velikosti. Daší vylepšením je zakomponování již zmiňovaného přeskakování sníkmu, kde sledovaný objekt chybí. Pro korekci chyby Kalmanova filtru se pak používá prediktovaná hodnota. To by mělo mít z podstaty Kalmanova filtru za následek, že nebude připočtena k měření v daném snímku žádná chyba.

6.4 Metody odečítání pozadí

Kromě Mixture of Gaussian, jejíž funkce je součástí OpenCV, jsou zde implementovány všechny ostatní metody odečítání pozadí. A to i s podpůrnými funkcemi a filtry pro konečnou úpravu FG modelu.

Nejprve k okrajové diferenční metodě. Metodu řeší sada tří funkcí. První z nich je inicializační. S cílem alokovat místo pro aktuální a poslední snímek. Dále pak přiřadit těmto snímkům počáteční stejnou hodnotu. Druhá funkce `runSimpleFgBg` provádí samotný výpočet. Postupně se prochází aktuální snímek pixel po pixlu. Pro každý pixel se pak berou hodnoty aktuálního a minulé. Ty se odečítají a výsledek se pak prahuje. Podle toho se určí, zda je či není daný pixel součástí FG modelu. Poslední funkce pak už jen maže alokovaná pole při ukončování programu.

O metody odečítání pozadí se stará 5 funkcí, z toho 2 řeší následné filtrování. Inicializační funkce zde volá konstruktor třídy Daubechies a převádí vstupní obraz na šedotónovou reprezentaci. Funkce `addWavFrames`, která řeší přídávání dalších obrazů než se dosáhne požadované délky sekvence, také pouze převádí obraz na šedotónový. Nejduležitější funkci tady je `runWavFilter`, která provádí požadované vlnkové filtrace. Funkce prochází obraz pixel po pixelu. U každého pixelu se vezmou hodnoty daného pixelu ze sekvence snímků požadované délky a předloží se filtrovací metodě, která je obsažena v třídě Daubechies. Pak se na daném pixelu posune celá sekvence a výsledek vlnkového filtru je uložen do aktuálního uvolněného snímku, v důsledku posunu sekvence. Pak se volá funkce `treshold_estimate`, která nastavuje lokálně aktuální práh. Výstupem je matice s lokálními prahy pro jednotlivé oblasti. Tato matice jde do další funkce `convol_enrg`, která pomocí konvoluce vyhlazuje obraz, který i následně prahuje. Obraz je na závěr doupraven morfologickými operacemi binárního uzavření a otevření. Pro operace dilatace a eroze používám funkce obsažené v OpenCV. Postup je takový, že se nejprve provede eroze pro upravený strukturní element velikosti 2x2. Dále se provádí 4 dilatace s předdefinovaným strukturním elementem 3x3, pak třikrát eroze se stejným strukturním elementem a na závěr opět dilatace s upraveným strukturním elemenetm 2x2.

Co se týče funkce `treshold_estimate`, tak tato funkce dělí na pevno obraz na oblasti o velikosti 40x40 pixelů. V každé oblasti zvlášť pak vytvářím histogram jednotlivých úrovní jasu. Určení prahu pak spočívá v tom, že vezmu celkový počet pixelu v oblasti. Pro 40x40 to je 400. Potom sčítám od nulové úrovně jasu jednotlivé úrovně jasu a na kolika pixelých jsou. Když překročím hodnotu 96% celkového počtu pixelů v oblasti, tak k hodnotě jasu, která toho docílila, přičtu další 4 úrovně jasu. Výslednou úroveň jasu pak beru jako práh pro danou oblast. Všechny v tom to odstavci pospané proměnné jsou stanovené na pevno a vzájemně na sebe navazují. Proto nemůžu vstupní obrazy o různém rozlišení dělit na vždy stejný počet oblastí. Měl bych tak různě veliké oblasti. Každá oblast přitom vyžaduje specifické vlastní nastavení dalších proměnných.

Funkce `convol_enrg` průměruje energie jednotlivých pixelů. Využívá přitom konvoluce a konvolučního jádra

$$h_8 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

kde konvoluci na krajích obrázku ošetřuji tak, že se používá jen část konvolučního jádra. Na závěr konvoluce každého pixelu pak každý pixel prahuji pomocí patřičného lokálního prahu.

6.5 Třída s vlnkovými filtry

Třída Daubechies obsahuje všech pět používaných vlnkových filtrů. Jde o *Daubechies 4* (D4), *Daubechies 6* (D6), *Daubechies 8* (D8), *Coiflet 6* a *Legendre 6*. Číslovky za názvem označují délku

vlnky, tedy počet užitých koeficientů a délku sekvence snímků. Všechny tyto filtry fungují na stejném principu. Rozdílem je, že filtry používají vlnky rozdílných délek a jejich velikostní a vlnkové transformační koeficienty jsou také rozdílné. Základem je jedna veřejná metoda, z které se volají ostatní soukromé metody jednotlivých specifických filtrů.

Myšlenku výpočtu vlnkového filtru nad sekvencí snímku beru z [30], kde je popsán algoritmus výpočtu. Každý krok vlnkové transformace aplikuje nejprve velikostní funkci na datový vstup. Mějme na vstupu N hodnot, potom bude velikostní funkce aplikována ve vlnkovém transformačním kroku na N/2 vyhlazovaných hodnot. V pořadí provádění vlnkové transformace jsou pak tyto vyhlazené hodnoty uloženy ve spodní polovině N hodnotového vstupního vektoru.

Každý krok vlnkové transformace taky aplikuje vlnkovou funkci na vstup. Vlnková funkce je podobně jako velikostní funkce aplikována N/2 hodnot k výpočtu rozdílu změn v datech. Její hodnoty se pak ve směru vlnkové transformace ukládají v horní polovině vstupního vektoru.

Velikostní funkce je v případě D4

$$a_i = h_0 s_{2i} + h_1 s_{2i+1} + h_2 s_{2i+2} + h_3 s_{2i+3}$$
$$a[i] = h_0 s[2i] + h_1 s[2i+1] + h_2 s[2i+2] + h_3 s[2i+3]$$

a vlnková funkce D4

$$c_i = g_0 s_{2i} + g_1 s_{2i+1} + g_2 s_{2i+2} + g_3 s_{2i+3}$$
$$c[i] = g_0 s[2i] + g_1 s[2i+1] + g_2 s[2i+2] + g_3 s[2i+3]$$

Každá iterace pak tady představuje jeden krok vlnkové transformace. Index *i* se vždy inkrementuje o dvě.

Pro potřeby mého algoritmu pak nepotřebuji počítat transformaci pro cely vstupní datový vektor. Mě stačí pouze spočítat poslední hodnotu, která je nejvýraznější diferencí datového vstupu, a tedy určuje pohyb v obraze. Místo dvou rovnic pak počítám pouze

$$c_3 = g_0 s_2 + g_1 s_3 + g_2 s_0 + g_3 s_1,$$

kde c_3 je výsledná hodnota filtrace, která je výstupem posledního kroku vlnkové transformace pro daný pixel. *s* je vektor hodnot, který představují sekvenci hodnot posledních 4 snímku (v případě užití D4, kdy se používá Daubechiesové vlnka o délce čtyři) v daném pixelu. A g je vektor vlnkových koeficientů. Ty se odvozují z velikostních koeficientů, tak že se k nim velikostní koeficienty přiřadí v opačném pořadí a každý druhý koeficient je vynásoben mínus jedničkou. Jednotlivé vlnkové koeficienty jsou konstanty, které jsou ve třídě napevno definovány.

7 Experimentální výsledky

Jak už jsem popisoval v předchozí kapitole, tak o testování aplikace se stará třída *tests*. Zde jsem naimplementoval čtyři různé testy. Dva z nich jsou zaměřené na přesnost sledování. Třetí počítá průchod objektů určitým místem a čtvrtý pak aplikuje na průjezdové místo objektu klasifikační metodu *Precision and Recal*. K této třídě jsem pak navrhl dva programy, které jsou určené k anotaci testovaných videí. Každý na jeden typ testů.

Třída tests

Třídu používá *main*. Její konstruktor dostává na základě typu testu, jména videa a typu metody pro odečítání pozadí jméno s daty anotačního souboru a jméno souboru, kam budou uloženy výsledky. Další komunikační vazbou je veřejná metoda *testing*, kde probíhají testy. Metoda si bere z aplikace seznam blobů, které jsou sledovány a ty pak podrobuje jednotlivým testům. Některé testy se vypisují do výsledného souboru již průběžně, ale všechny výsledky testování jsou finálně zpracovány další veřejnou třídou *result*, která je volána před ukončením aplikace.

7.1 Testy zaměřené na přesnost

Tyto testy testují dvě věci. Zaprvé jak dlouho od detekce je aplikace schopná daný objekt testovat. A za druhé s jakou přesností od správné pozice. U testování, jak dlouho algoritmu dovede sledovat pohybující se objekt, se do výsledku zaznamenává, na kolikátém kontrolním bodu bylo přerušeno sledování objektu, ať z důvodů zmizení blobu a zaniknutí sledovacího Kalmanova filtu, nebo z důvodů nepřesného sledování, tedy špatné predikce budoucí pozice sledovaného objektu. Kontrolní body, které určují skutečnou pozici pohybujícího se objektu, se vytvářejí v anotačním programu. Další věc, která se sleduje u typu videí s objekty, které mají silně náhodný pohyb, je kolikrát z kolika pokusů se detekční a sledovací algoritmus trefil do kontrolních bodů.

U testování přesnosti sledování správné pozice objektu se na výsledku testu zaznamenávají odchylky v jednotlivých kontrolních bodech od reálné pozice. Vzdálenost je dána v pixelech. Na závěr je pak tato odchylka od reálu zprůměrovaná.

U prvního testu se rozhoduje o tom, zda je či není sledovaný objekt ztracen, podle velikosti odchylky od reálné pozice. Jako práh se bere detekovaná velikost blobu, ze které se počítá úhlopříčka. Úhlopříčka vynásobená hodnotou 0.8 (správně by to mělo být 0.5) je pak použita jako práh odchylky.

Tyto testy testují dvě podobné třídy videí. První třídou jsou krátká videa, kdy se vždy v interiéru pohybuje jedno autíčko. Pohyb autíčka přitom není zcela přímý. Druhá třída videi sleduje

v interiéru dlouhý zcela náhodný pohyb vrtulníčku. U této druhé třídy se právě využívá při testování pouze testu, kde se určuje, kolikrát aplikace správně rozpoznala objekt v kontrolním bodu.

Výsledky testů videí s autíčky

Jak jsem již psal, zde se testuje přesnost sledování trajektorie autíček v krátkých sekvencích. Kontrolní body jsou umístěny každých 5 snímků. Porovnání jednotlivých metod je v Tabulkách 1-6. U těchto typů videa natočených web kamerou Unibrain Fire-i se ukázala zcela nefunkční diferenční metoda. Pozadí se zde odečítá pouze každý druhý snímek. To má za následek, že se vůbec neinicializuje sledování.

Sloupce v tabulkách mají tento význam. **USKBč**. je *udržení sledování do kontrolního bodu č.*; **% USKBč.** je procentuálním vyjádřením předchozího; **celkem KB** je celkový *počet kontrolních bodů v anotovaném videu*; **OK KB** *počet správných detekcí v kontrolním bodě*; **% OK** jeho procentuálním vyjádřením; **n OK KB** je *počtem špatných detekcí v kontrolním bodě*; **accurate** je *průměrnou odchylkou ve všech kontrolních bodech.*

auticko 1

metoda	USKBč.	% USKBč.	celkem KB	OK KB	% OK	n OK KB	accurate
D4	2	11,76%	17	12	70,59%	5	8,3125
D6, Coif, Legr	8	47,06%	17	13	76,47%	4	11,9545
D8	8	47,06%	17	11	64,71%	6	17,6364
diference	1	5,88%	17	0	0,00%	17	NDF
Mix of Gauss	8	47,06%	17	13	76,47%	4	195,746

Tabulka 1 Výsledek testu pro video auticko1.

auticko 2

metoda	USKBč.	% USKBč.	celkem KB	OK KB	% OK	n OK KB	accurate
D4	2	11,76%	17	11	64,71%	6	13,1579
D6, Coif, Legr	4	23,53%	17	10	58,82%	7	15
D8	5	29,41%	17	10	58,82%	7	16,6087
diference	1	5,88%	17	0	0,00%	17	NDF
Mix of Gauss	3	17,65%	17	10	58,82%	7	49,2174

Tabulka 2 Výsledek testu pro video auticko2.

auticko 3

metoda	USKBč.	% USKBč.	celkem KB	OK KB	% OK	n OK KB	accurate
D4	13	100,00%	13	13	100,00%	0	10,8182
D6, Coif, Legr	1	7,69%	13	8	61,54%	5	11,4545
D8	1	7,69%	13	9	69,23%	4	10,2143
diference	1	7,69%	13	0	0,00%	13	NDF
Mix of Gauss	1	7,69%	13	9	69,23%	4	8,4

Tabulka 3 Výsledek testu pro video auticko3.

auticko 4

metoda	USKBč.	% USKBč.	celkem KB	OK KB	% OK	n OK KB	accurate
D4	5	25,00%	20	12	60,00%	8	8,29412
D6, Coif, Legr	6	30,00%	20	14	70,00%	6	9,05263
D8	6	30,00%	20	15	75,00%	5	10,4286
diference	1	5,00%	20	0	0,00%	20	NDF
Mix of Gauss	7	35,00%	20	16	80,00%	4	9,7619

Tabulka 4 Výsledek testu pro video auticko4.

auticko 5

metoda	USKBč.	% USKBč.	celkem KB	OK KB	% OK	n OK KB	accurate
D4	8	53,33%	15	13	86,67%	2	5,66667
D6, Coif, Legr	8	53,33%	15	13	86,67%	2	9,94444
D8	8	53,33%	15	14	93,33%	1	16,1429
diference	1	6,67%	15	0	0,00%	15	NDF
Mix of Gauss	8	53,33%	15	13	86,67%	2	4,86667

Tabulka 5 Výsledek testu pro video auticko5.

auticko 6

metoda	USKBč.	% USKBč.	celkem KB	OK KB	% OK	n OK KB	accurate
D4	43	100,00%	43	43	100,00%	0	4,95455
D6, Coif, Legr	1	2,33%	43	41	95,35%	2	7,2
D8	43	100,00%	43	43	100,00%	0	13,5238
diference	1	2,33%	43	0	0,00%	43	NDF
Mix of Gauss	38	88,37%	43	41	95,35%	2	372,341

Tabulka 6 Výsledek testu pro video auticko6.

Nejdůležitější věc, kterou tyto, ale i další testy ukázaly, je fakt, že metody, které využívající k odečtu pozadí Daubechiesové, Coifletový a Legendrovy vlnky, jsou ve výsledku stejné. Z toho plyne, že na samotných hodnotách rozměrového a vlnkového koeficientu ani tolik nezáleží. Důležitá je délka vlnky. Tyto výsledky ukázaly, že metoda Mixture of Gaussian sice celkově nejpřesněji detekuje objekty vzhledem ke kontrolním bodům. Její nevýhodou ale je, že ze šumu a změn osvětlení kromě toho generuje velké množství falešných objektů. Důkazem toho je vysoká hodnota *accurate*, která v určených kontrolních bodech počítá průměrnou přesnost všech sledovaných objektů, tedy i velkého množství falešných a vzdálených „objektů“. Z ostatních metod si nejlépe vede D4, která co do rychlosti jako jediná konkuruje metodě Mixture of Gaussian (když se nepočítá jednoduchá diference).

Výsledky testů videí s helikoptérou

Dalším typem sledovaných videí je model helikoptéry, který se náhodně vznáší v místnosti. Jde o extrém, s kterým si snaží poradit zejména Kalmanův filtr. Ve videu jsou rozmístěny kontrolní body každých 20 snímků. Významy jmen sloupců Tabulek 7 a 8 jsou pak obdobné jako v předchozím případě.

helikoptera 1

metoda	USKBč.	% USKBč.	celkem KB	OK KB	% OK	n OK KB
D4	14	32,56%	43	31	72,09%	12
D6, Coif, Legr	12	27,91%	43	32	74,42%	11
D8	5	11,63%	43	38	88,37%	5
Mix of Gauss	11	25,58%	43	38	88,37%	5

Tabulka 7 Výsledek sledování náhodného pohybu ve videu helicopter1.

helikoptera 3

metoda	USKBč.	% USKBč.	celkem KB	OK KB	% OK	n OK KB
D4	8	11,59%	69	59	85,51%	10
D6, Coif, Legr	5	7,25%	69	59	85,51%	10
D8	6	8,70%	69	62	89,86%	7
Mix of Gauss	4	5,80%	69	51	73,91%	18

Tabulka 8 Výsledek sledování náhodného pohybu ve videu helicopter3.

Zde jsou metody s délkou vlnky 6 průměrné a nevybočují. Relativně dobře se také vede D4. Na Obrázku 7.1 je ukázka z testování.

Obrázek 7.1 Sledování helikoptéry ve videu helicopter1.

7.2 Testy založené na výskytu objektu v určitém místě.

Zde se testuje průchod sledovaných objektů určeným místem v obraze. První test jednoduše počítá počet objektů, které projdou daným místem obrazu. Druhým testem pak je testování přesnosti aplikace pomocí statistického klasifikátoru *Precision and Recall*. Tato přesnost je pět testována vzhledem k průchodu sledovaných objektů určitým místem obrazu.

Při spouštění anotačního programu se nejprve nastavuje rychlost přehrávání anotovaného videa. V samotné aplikaci se pak na začátku nastaví v obraze oblast zájmu, vzhledem ke které se bude testovat přítomnost procházejících blobů. Tato oblast je v uživatelském zobrazení zvýrazněna. Pak se sleduje průchod objektů oblastí a na základě přítomnosti nebo nepřítomnosti objektů se dá vědět aplikaci pomocí stisku příslušné klávesy.

Třídou videí, které zde testuji, jsou videa v přírodě, která snímají z větší vzdálenosti silniční provoz (minimálně ze 150 metrů). Snahou je využívat videa, kde se neplete před silnicí velké množství stromů.

Precision and Recall

Precision a Recall jsou statistické klasifikátory. Precision vyjadřuje pravděpodobnost, že libovolná obdržená hodnota je relevantní. Recall je pravděpodobnost, že libovolná relevantní hodnota je obdržena. Další termíny, které se v kontextu klasifikátorů používají, jsou *true positives*, *true negatives*, *false positives* a *false negatives*. Používají se k porovnávání dané klasifikace vzhledem k správné vzorové klasifikaci (jako v mém případě, výsledky sledování vzhledem k anotovanému vzoru). Význam termínů popisuje Obrázek 7.1. V mém případě klasifikace *E1* a *E2* představují, zda v daném místě v daný čas objekt je (E1) nebo není (E2).

		correct result / classification	
		E1	E2
obtained result / classification	E1	tp (true positive)	fp (false positive)
	E2	fn (false negative)	tn (true negative)

Obrázek 7.2 Význam termínů true positive, true negativ, false positive, false negativ [].

Podle toho jsou pak Precision a Recall definováný pomocí výrazů

$$Precision = \frac{tp}{tp + fp}$$

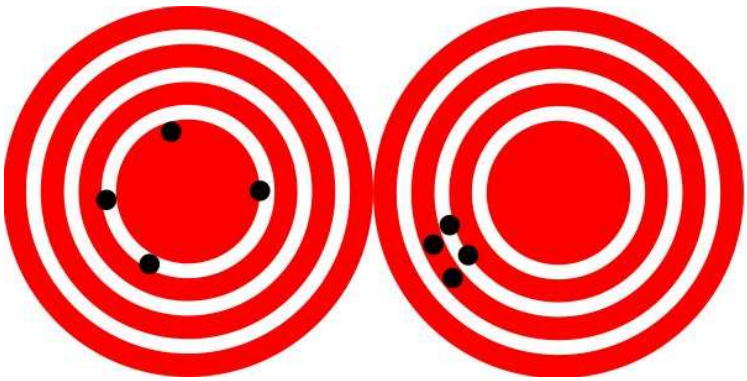
$$Recall = \frac{tp}{tp + fn}$$

Někdy se Recall nazývá také jako *True Positive Rate*. Podle toho jsou pak definovaná další používaná měření, *True Negativ Rate* a *Accuracy* (Přesnost):

$$True\ Negativ\ Rate = \frac{tn}{tn + fp}$$

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

Pěkně je pak v [] vysvětlený rozdíl mezi Accuracy a Precision. V analogii terče a šípů je Accuracy tím větší čím jsou šípy co nejblíže středu. Naproti tomu Precision je tím, větší čím menší shluk na terči šípy vytvoří (Obrázek 7.3.).



Obrázek 7.3 Vlevo je vysoký Accuracy a nízký Precision, vpravo tomu je naopak [].

Test počtu projetých objektů

Zde testují schopnosti aplikace přesně počítat projetá vozidla určeným místem. To je zobrazeno v Tabulkách 9-11. Význam sloupců je následující: **anot. obj.** je počet vozidel, která byla spočítána při anotaci; **spoč. obj.** je počet vozidel napočítaný aplikací; **odchylka** je absolutní rozdíl dvou předchozích hodnot; **% odch.** je její procentuální vyjádření.

auto 2

metoda	anot. obj.	spoč. obj.	odchylka	% odch.
D4	65	71	6	9,23%
D6, Coif, Legr	65	80	15	23,08%
D8	65	77	12	18,46%
Mix of Gauss	65	64	1	1,54%
diference	65	45	20	30,77%

Tabulka 9 Výsledek testu videa autaUp2.

auto 3

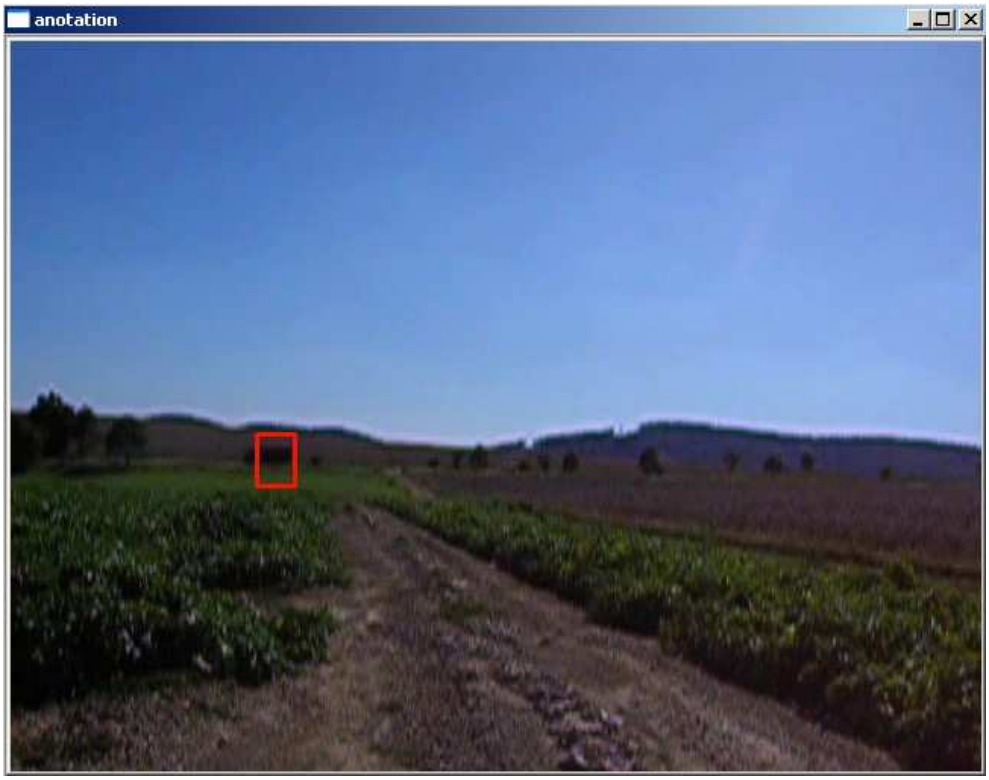
metoda	anot. obj.	spoč. obj.	odchylka	% odch.
D4	24	28	4	16,67%
D6, Coif, Legr	24	32	8	33,33%
D8	24	36	12	50,00%
Mix of Gauss	24	31	7	29,17%
diference	24	34	10	41,67%

Tabulka 10 Výsledek testu videa autaUp3.

auto 4				
metoda	**anot. obj.**	**spoč. obj.**	**odchylka**	**% odch.**
D4	53	52	1	1,89%
D6, Coif, Legr	53	60	7	13,21%
D8	53	56	3	5,66%
Mix of Gauss	53	53	0	0,00%
diference	53	54	1	1,89%

Tabulka 11 Výsledek testu videa autaUp4.

Zde dosahuje ve dvou ze tří případů nejlepších výsledků metoda Mixture of Gaussian. Její velkou nevýhodou pro reálné použití však je skutečnost, že metoda velmi špatně reaguje na šum způsobený změnou intenzity světla. S tím metody založené na vlnkové transformaci nemají problém. Opět i zde si vede velmi dobře metoda D4. Oblasti, jejichž průjezd se měří, jsou na Obrázcích 7.4, 7.5.a 7.6.



Obrázek 7.4 Zkoumaná oblast u videa autaUp2.

Obrázek 7.5 Zkumaná oblast u videa autaUp3.

Obrázek 7.6 Zkumaná oblast u videa autaUp4.

Precision a Recall

V Tabulkách 12 a 13 je zobrazen výsledek tohoto testu. **TP** je true positive, **FP** false positive, **TN** true negativ, **FN**, false negativ. **TRUE NG RT** pak znamená True Negative Rate.

auto 4

metoda	TP	FP	TN	FN	PRECISION	RECALL	TRUE NG RT	ACCURACY
D4	44	8	52	9	0,8462	0,8302	0,8667	0,8496
D6, Coif, Legr	47	13	50	6	0,7833	0,8868	0,7937	0,8362
D8	45	11	46	8	0,8036	0,8491	0,807	0,8273
diference	43	11	52	10	0,7963	0,8112	0,8254	0,819
Mix of Gauss	49	4	52	4	0,9245	0,9245	0,9286	0,9266

Tabulka 12 Výsledek testu Preciosn a Recall pro video autaUp4.

auto 3

metoda	TP	FP	TN	FN	PRECISION	RECALL	TRUE NG RT	ACCURACY
D4	15	13	17	9	0,5357	0,625	0,5667	0,5926
D6, Coif, Legr	11	21	17	13	0,3438	0,4583	0,4474	0,4516
D8	14	22	17	10	0,3889	0,5833	0,4359	0,4921
diference	20	14	17	4	0,5882	0,8333	0,5484	0,6727
Mix of Gauss	23	8	17	1	0,7419	0,9583	0,68	0,8163

Tabulka 13 Výsledek testu Preciosn a Recall pro video autaUp3.

Zde dosahuje nejlepšího výsledku metoda Mixture of Gaussian. V tomto testu si vede překvapivě dobře i metoda prosté diference dvou obrázku. Jako ve všech ostatních testech nejhůře dopadly metody s délkou vlnky 6 (D6, Coiflet, Legrend).

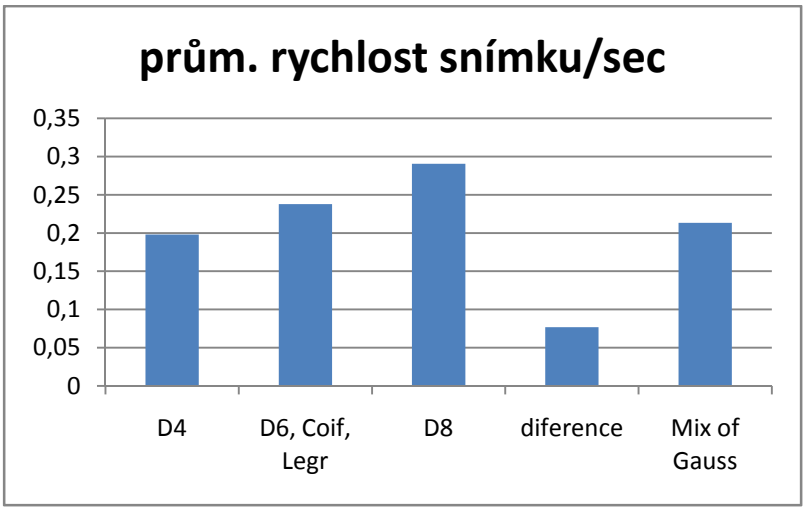
7.3 Test rychlosti

Na závěr testování jsem ještě provedl test, který zjišťuje rychlost algoritmu v závislosti na použité metodě (Tabulka 14)

rychlost metod (videa s rozlišením 640x480, 15fps)

metoda	auta 4	helikoptera 1	auticko 3	auticko 6
délka videa	325	59	6	12
D4	704	131	23	44
D6, Coif, Legr	849	157	28	52
D8	1037	191	34	64
diference	265	52	9	17
Mix of Gauss	720	133	25	50

Tabulka 14 Rychlost metod pro různá videa.



Tabulka 15 Graf ukazující rychlost metod na jeden snímek.

Z grafu a tabulky vyplívá, že nejrychlejší je očekávaně diferenční metoda. Druhá je pak metoda D4 a těsně v závěsu je Mixture of Gaussian. Z tabulky dále vyplívá, že čím je zpracovávané video delší, tím je celková rychlost na jeden snímek vyšší.

8 Závěr

Tato diplomová práce nejprve shrnuje důležitou teorii a základní metody, funkce a postupy, které jsou nezbytné pro valnou část metod detekce a sledování pohyblivých objektů ze statické kamery.

V práci jsem nejprve uvedl některé základní teoretické poznatky, které jsou nezbytné z důvodu proniknutí do problematiky práce. Začal jsem od Obrazové funkce přes filtry, konvoluce, morfologické operace a pro tuto práci důležité vlnky. Tyto poznatky jsem dále rozvinul i konkrétními příklady funkčnosti. Ukázal jsem některé dílčí používané metody, u kterých jsem naznačil, jak fungují. Popsal jsem zde Bayesovskou filtraci a do ní patřící Kalmanův filtr. Popsal jsem, jak funguje predikce budoucích stavů pomocí Kalmanova filtru. Dále jsem navrhl a popsal, jaké typy videí má cílová aplikace přibližně řešit. Pak jsem navrhl a implementoval algoritmus, který řeší daný problém. K odečítání pozadí jsem zde využil Vlnkové filtrace a k sledování detekovaných objektů právě Kalmanův filtr.

U metody Vlnkové filtrace jsem se musel potýkat s tím, abych v obraze co nejvíce eliminoval pohyb a abych mohl nastavovat automaticky práh na prahování obrazu vycházejícího z Vlnkové filtrace. Problém eliminace nežádoucích pohybů jsem vyřešil rozdělením obrazu do malých oblastí, pro které samostatně nastavuji hodnotu prahu. Ten je vypočítáván z histogramu jasů v dané oblasti.

Pro fázi experimentů jsem navrhl anotační programy na daná vstupní videa. Výstup pak byly data používaná testovací třídou. K testování jsem navrhl čtyři typy testů. 2 jsou spíše pro videa z interiéru a druhé dva jsou určeny ke sledování silničního provozu na vozovce z větší dálky. Výsledkem testů je, že mé navržené metody dosahuji srovnatelných výsledků jako jiné pokročilé aplikace tohoto typu. Pokud jde o rychlost, tak i s využitím nejrychlejší metody odečítání pozadí (D4) nedosahuji real-timové rychlosti pro videa s rozlišením 640x480 a 15fps. Pro toto rozlišení je aplikace asi 2x pomalejší. Tedy pro rozlišení 480x320 by už mělo být vše v pořádku. Aplikace si také umí poradit celkem dobře s méně nepříznivými podmínkami. Šum není problém, občas pouze aplikaci vadí výraznější pohyb blízké trávy či stromů.

Mým osobním přínosem v této práci je hlavně systém detekce prahu, využívaného při tvorbě pohybujícího se popředí (FG modelu). Dále to jsou různé drobnosti a vylepšení jako je poskládání pořadí matematických morfologických operací eroze a dilatace, které se aplikují na téměř hotový FG model.

Co se týče dalšího možného vývoje aplikace, tak to je rozhodně optimalizace algoritmu, tak aby se zvýšila rychlost. Dále by bylo žádoucí vymyslet a přidat další typy testů. K aplikaci by také mohlo být navrženo nějaké pokročilé grafické rozhraní. Nyní se využívá základních možností, které nabízí OpenCV. Samozřejmě další alternativní metody jak detekcí tak na sledování objektů jsou též vítané. Zajímavé by pak bylo upravit aplikaci, tak aby zvládala úspěšně detekovat a sledovat i další třídy videí. Aby se dala aplikace využít všude, tak by to byla letadla na nebi a lodě na moři.

Literatura

[1] REULKE, R., et al. Traffic Surveillance using Multi-Camera Detection and Multi-Target Tracking. In *Proceedings of Image and Vision Computing New Zealand 2007*. Hamilton, New Zealand : [s.n.], 2007. s. 175-180. Dostupný z WWW: <digital.liby.waikato.ac.nz/conferences /.../ivcnz07-paper33.pdf>.

[2] *Frequency probability* [online]. [2005?] , 20 June 2009 [cit. 2009-07-29]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Frequentist>.

[3] *Bayesian probability* [online]. [2006?] , 28 July 2009 [cit. 2009-07-29]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Bayesian_probability>.

[4] *Bayes‘ theorem* [online]. [2006?] , 28 July 2009 [cit. 2009-07-30]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Bayes%27_theorem>.

[5] STEVENS, Michael. *Open Source Bayesian Filtering Classes* [online]. c2005 [cit. 2009-01-05]. Dostupný z WWW: <http://bayesclasses.sourceforge.net/Bayes++.html>.

[6] *Bayesian inference* [online]. [2007?] , 18 July 2009 [cit. 2009-07-25]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Bayesian_inference>.

[7] *Cascade algorithm* [online]. [2009?] , 30 May 2009 [cit. 2009-07-30]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Cascade_algorithm>.

[8] ZEMČÍK, Pavel. *Chyby v obraze, typy šumu, restaurace obrazu a optimální filtrace*. Brno : Ústav počítačové grafiky a multimédií, FIT VUT, [2008?]. 46 s. Přednáška Zpracování obrazu.

[9] WELCH, Greg, BISHOP, Gary. *An Introduction to the Kalman Filter* [online]. Department of Computer Science, University of North Carolina at Chapel Hill, 2006 , July 24, 2006 [cit. 2009-01-02]. Dostupný z WWW: <http://www.cs.unc.edu/~welch/media/pdf/kalman _intro.pdf>.

[10] VAN KEMPEN, Erik. *GeekBlog.nl* [online]. c2007 , February 9, 2008 [cit. 2009-01-04]. Dostupný z WWW: <http://geekblog.nl/>.

[11] *Blob detection* [online]. [2004?] , 17 May 2009 [cit. 2009-01-03]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Blob_detection>.

[12] *Video tracking* [online]. [2005] , 27 February 2009 [cit. 2009-01-04]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Video_trackingVideo tracking>.

[13] DEANE, Sarah. *A Comparison of Background Subtraction Techniques*. [s.l.] : University of Southampton, 2007. 15 s. Referát. Dostupný z WWW: <http://portfolio.ecs.soton.ac.uk/20/3/irp_report_ieee.pdf>.

[14] *Wavelet* [online]. [2003] , 18 July 2009 [cit. 2009-07-22]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Wavelet>.

[15] Davies D., Palmer P., Mirmehdi. *Detection and Tracking of Very Small Low Contrast Objects.*

Proceedings of the 9th British Machine Vision Conference, Sept. 1998,
URL: www.bmva.org/bmvc/1998/pdf/p135.pdf

[16] YILMAZ, Apler, JAVED, Omar, SHAH, Mubarak. Object tracking: A survey. In *ACM Comput. Surv.*. 38th edition. New York, USA : ACM, c2006. 13. 45 s. Dostupný z WWW: <http://portal.acm.org/citation.cfm?doid=1177352.1177355>.

[17] *Soubor:Konvoluce 2rozm diskretni.jpg* [online]. 2006 , 20 July 2006 [cit. 2009-07-24] .Dostupný z WWW: <http://cs.wikipedia.org/wiki/Soubor:Konvoluce_2rozm_diskretni.jpg>.

[18] *7.2. Konvoluční matice* [online]. c2007 [cit. 2009-07-24]. Dostupný z WWW: <http://docs.gimp.org/2.2/cs/plug-in-convmatrix.html>.

[19] *Mathematical morphology* [online]. 12 July 2009 [cit. 2009-07-24]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Mathematical_morphology>.

[20] HLAVÁČ, Václav. *Matematická morfologie*. Praha : ČVUT, [2003?]. 41 s. Přednáška. Dostupný z WWW: <cmp.felk.cvut.cz/~hlavac/.../71-03BinMatMorfolCesky.pdf>.

[21] *Legendre wavelet* [online]. [2004?] , 5 July 2009 [cit. 2009-07-30]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Legendre_wavelet>.

[22] *Haar wavelet* [online]. [2003] , 18 June 2009 [cit. 2009-07-29]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Haar_wavelet>.

[23] *Daubechies wavelet* [online]. [2004] , 25 June 2009 [cit. 2009-07-30]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Daubechies_wavelet>.

[24] *Coiflet* [online]. [2008] , 6 May 2009 [cit. 2009-07-30]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Coiflet>.

[25] NASHAN, Liu, EN-BING, Lin. Legendre Wavelet Method for Numerical Solutions of Partial Differential Equations. *Wiley InterScience* [online]. 2008 [cit. 2009-07-30]. Dostupný z WWW: <http://www3.interscience.wiley.com/cgi-bin/fulltext/121639785/PDFSTART>.

[26] STOLLNITZ , Eric, DEROSE, Tony, SALESIN, David. Wavelets for computer graphics: A primer, part 1. *IEEE Computer Graphics and Applications* [online]. 1995 [cit. 2009-07-25], s. 8. Dostupný z WWW: <http://grail.cs.washington.edu/projects/wavelets/article/wavelet1.pdf>.

[27] VAN KEMPEN, Erik. *Blob detection V: growing regions algorithm* [online]. 2008 , January 14, 2008 [cit. 2009-01-04]. Dostupný z WWW: <http://geekblog.nl/entry/24>.

[28] VAN KEMPEN, Erik. *Blob detection* [online]. 2007 , April 8, 2007 [cit. 2009-01-04]. Dostupný z WWW: <http://geekblog.nl/entry/12>.

[29] JINZI, Mao. *Tracking a tennis ball using image processing techniques*. Saskatoon : University of Saskatchewan, c2006. 124 s. Diplomová práce. Dostupný z WWW: <http://library2.usask.ca/theses/available/etd-08302006-125935/unrestricted/thesis_Jinzi_Mao.pdf>.

[30] KAPLAN, Ian. *The Daubechies D4 Wavelet Transform* [online]. 2001 , January 2002 [cit. 2009-08-01]. Dostupný z WWW: <http://www.bearcave.com/misl/misl_tech/wavelets/daubechies/index.html>.

[31] *Precision and recall* [online]. [2005] , 11 July 2009 [cit. 2009-08-04]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Precision_and_recall>.

[32] *Accuracy and precision* [online]. 2006 , 2 August 2009 [cit. 2009-08-03]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Accuracy_and_precision#Accuracy_in_binary_classification>.

[33] ŽÁRA, Jiří, et al. *Moderní počítačovágrafika*. Brno : Computer Press, 2004. 609 s. ISBN 80-251-0454-0.

[34] YANG, Changjiang, DURAISWAMI, Ramani, DAVIS, Larry. Fast Multiple Object Tracking via a Hierarchical Particle Filter. *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference* [online]. 2005, no. 1 [cit. 2009-08-04], s. 212-219. Dostupný z WWW: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&isnumber=&arnumber=1541259>. ISSN 1550-5499.

[35] FLEURET, François, SAHBI, Hichem. Coarse-to-Fine Object Detection. *Ercim news : online edition* [online]. 2003, is. 55 [cit. 2009-08-04]. Dostupný z WWW: <http://www.ercim.org/publication/Ercim_News/enw55/fleuret.html>.

9 Seznam příloh

Příloha A …….. Manuál programu

Příloha B …….. Plakát prezentující práci

Příloha C …….. Sada testovacích videí a spouštěcí skripty (na DVD)

Příloha D …….. Původní výsledky testů (na DVD)

Příloha E …….. Zdrojové texty (na DVD)

Příloha F …….. DVD

Příloha A Manuál programu

Samotným program obsahuje pouze možnost okamžitého ukončení pomocí klávesy ESC. Jinak se chová jako dávka. Při spuštění se zadávají všechny potřebné parametry. Program vyžaduje minimálně jeden povinný parametr. Ostatní parametry jsou volitelné. Jejich proměnné jsou v programu přednastavené.

Povinné parametry (jeden z nich musí být minimálně zadán):

-h	parametr vyžadující nápovědu (zadává se samostatně;
[…] | input.avi	jméno souboru se vstupním videem.

Ostatní volitelné parametry:

-out	**[…] |** output.avi	Jméno souboru, kam se ukládá výstupní video (nastaveno na out.avi).
-n1	(0.0, 1.0)	Číselná hodnota v rozsahu (0.0, 1.0) specifikující velikost okolí při hledání druhého blobu u inicializace Kalmanova filtru (0.3).
-n3	(0.0, 1.0)	Číselná hodnota v rozsahu (0.0, 1.0) specifikující velikost okolí při hledání třetího blobu u inicializace Kalmanova filtru (0.1).
-nk	(0.0, 1.0)	Číselná hodnota v rozsahu (0.0, 1.0) specifikující velikost okolí od predikovaného blobu při hledání naměřeného blobu u korekce Kalmanova filtru (0.1).
-method	<BG, 2COMP, WAV4, WAV6, WAV8, COIF, LEGR>	Parametr, který určuje užití metody detekce pohybu ve statickém obraze. BG je metoda Mixture of Gaussian, 2COMP diference dvou posledních snímků, WAV4 Daubechiesové vlnka délky 4, WAV6 Daubechiesové vlnka délky 6, WAV8 Daubechiesové vlnka délky 8, COIF Coifletova vlnka délky 6 a LEGR je Legendrova vlnka délky 6.
-test	< T_TRACK, T_ACCUR, T_COUNT, T_RCL_PRS>	Parametr určující typ použitého testu. T_TRACK je testování jak dlouho může být úspěšně sledován pohyb objektu. T_ACCUR pak testuje přesnost takového sledování. T_COUNT počítá počet objektů, které projdou daným místem. T_RCL_PRS pak pro dané místo počítá Precision, Recall, Accuracy a True Negativ Rate.

Ukázka použití parametrů:

```
Det_track_1.2.exe input.avi -out output.avi -method WAV6 –test
T_RCL_PRS –nk 0.1
Det_track_1.2.exe input.avi -method COIF
```

Ovládání anotačních programů

K anotaci videí jsem vytvořil dva prográmky. Oba se spouštějí s jedním volitelným a jedním povinným parametrem. Povinný parametr je u obou stejný, jméno anotovaného videa. Druhým stejným parametrem je **–h** pro nápovědu.

Program s názvem **Anotace_videa_1** anotuje videa pro testy T_TRACK a T_ACCUR. Volitelný parametrem je číslo určující skok mezi určovanými kontrolními body ve videu (Nastaveno implicitně na 20). Číslo je v oboru celých čísel a určuje, o kolik snímků se skočí. V samotném programu se dále využívá myši. Tou se na jednotlivé objekty přesně kliká levým tlačítkem myši. Je tak vytvořen kontrolní bod. Pravým tlačítkem se pak daný kontrolní snímek přeskakuje, když není co anotovat

Program **Anotace_videa_2** anotuje videa pro testy T_COUNT a T_RCL_PRS. Volitelný parametr má charakter **–speed** *číslo*. Číslo je opět celé. Parametr určuje rychlost přehrávání videa (Implicitně 100). Čím vyšší hodnota, tím pomalejší přehrávání. V samotném programu se na začátku musí dvěma kliky levým tlačítkem myši v obraze určit oblast, kde se bude počítat průchod pohybujících se objektů. Potom se vždy při průchodu objektu stiskne klávesa “b“ a když je oblast prázdná, tak se občas stiskne “n“.

Obrázek Příloha A.0.1 Tak vypadá označená oblast v anotovaném videu, kde se počítá počet průchodu touto oblastí.